Titre: Title:	Développement d'algorithmes parallèles pour la simulation d'écoulements de fluides dans les milieux poreux
Auteur: Author:	David Jean-Emmanuel Vidal
Date:	2008
Type:	Mémoire ou thèse / Dissertation or Thesis
Référence: Citation:	Vidal, D. J.-E. (2008). Développement d'algorithmes parallèles pour la simulation d'écoulements de fluides dans les milieux poreux [Thèse de doctorat, École Polytechnique de Montréal]. PolyPublie. https://publications.polymtl.ca/8198/

UNIVERSITÉ DE MONTRÉAL

DÉVELOPPEMENT D'ALGORITHMES PARALLÈLES POUR LA SIMULATION D'ÉCOULEMENTS DE FLUIDES DANS LES MILIEUX POREUX

DAVID JEAN-EMMANUEL VIDAL
DÉPARTEMENT DE GÉNIE INFORMATIQUE ET GÉNIE LOGICIEL
ÉCOLE POLYTECHNIQUE DE MONTRÉAL

THÈSE PRÉSENTÉE EN VUE DE L'OBTENTION
DU DIPLÔME DE PHILOSOPHIÆ DOCTOR
(GÉNIE INFORMATIQUE)
DÉCEMBRE 2008

Your file Votre référence
ISBN: 978-0-494-48893-5
Our file Notre référence
ISBN: 978-0-494-48893-5

UNIVERSITÉ DE MONTRÉAL

ÉCOLE POLYTECHNIQUE DE MONTRÉAL

Cette thèse intitulée:

DÉVELOPPEMENT D'ALGORITHMES PARALLÈLES POUR LA SIMULATION D'ÉCOULEMENTS DE FLUIDES DANS LES MILIEUX POREUX

présentée par: VIDAL David Jean-Emmanuel
en vue de l'obtention du diplôme de: Philosophiæ Doctor
a été dûment acceptée par le jury d'examen constitué de:

M. GUIBAULT François, Ph.D., président
M. BERTRAND François, Ph.D., membre et directeur de recherche
M. ROY Robert, Ph.D., membre et codirecteur de recherche
M. HENICHE Mourad, Ph.D., membre
M. SÉNÉCHAL David, Ph.D., membre

À ma conjointe Karyne,

À mes parents,

REMERCIEMENTS

Je souhaite témoigner toute ma gratitude à mes directeur et co-directeur de thèse, Professeurs François Bertrand et Robert Roy, pour leurs encouragements, leur aide, leur disponibilité, leur professionnalisme et surtout leur patience au cours de ces longues années.

J'adresse aussi tous mes remerciements aux co-auteurs du troisième article, Docteurs Grégoire Pianet, Cathy Ridgway et Joachim Schoelkopf, pour leurs contributions respectives.

I would like to extend my gratitude to my successive program managers at FPInnovations-Paprican, Doctors Tetsu Uesaka and Xuejun Zou, for the opportunity that was given to me and their support.

Je tiens aussi à remercier mes collègues à FPInnovations-Paprican, Docteur François Drolet et Monsieur Christian Poirier, et à l'École Polytechnique, Monsieur Louis-Alexandre Leclaire, pour leur aide, leur soutien technique et/ou pour les nombreuses discussions utiles que nous avons eu ensemble.

Je remercie aussi les membres du jury, Professeurs François Guibault et David Sénéchal, et Docteur Mourad Heniche, pour avoir bien voulu faire parti du jury.

Mes remerciements vont aussi aux organisations suivantes, FPInnovations-Paprican, le Réseau Québécois de Calcul de Haute Performance et le réseau CRSNG Sentinel pour leur support.

Finalement, je tiens à remercier ma conjointe Karyne, toute ma famille et mes amis pour leur patience et leur support inestimables à travers les années.

RÉSUMÉ

Deux algorithmes parallèles différents basés sur la méthode de Boltzmann sur réseau (MBR) ont été conçus pour la simulation d'écoulement monophasique dans les milieux poreux complexes. Ils sont basés sur des algorithmes MBR efficaces du point de vue de la gestion de la mémoire, c'est-à-dire les algorithmes *one-lattice* and *shift*, combinés à une structure de données vectorielle, une décomposition de domaine par répartition équitable des nœuds fluides et des schémas de transfert de données optimaux. L'algorithme *shift* inclut aussi un schéma à relaxation unique unitaire qui permet une économie additionnelle de mémoire, mais limite sa validité aux cas de fluides newtoniens. Ils offrent une performance parallèle élevée en équilibrant la charge de calcul et en réduisant la quantité de données transférées entre les processeurs, et réduisent significativement l'utilisation de la mémoire comparé aux codes parallèles précédemment présentés dans la littérature. Des modèles théoriques de performance parallèle et d'utilisation de la mémoire développés dans le cadre de ces travaux montrent qu'ils procurent une bonne évolutivité. Des efficacités parallèles aussi grande que 79% pour des simulations contenant plusieurs milliards de nœuds fluides sur 128 processeurs sont rapportées. L'application d'un de ces algorithmes dans le cadre de la simulation de l'écoulement à travers des entassements compressés faits de particules sphériques polydisperses a démontré la précision et l'efficacité remarquables de l'algorithme proposé. Une corrélation de Carman-Kozeny modifiée prenant en compte le niveau de compression et la polydispersité des particules a pu ainsi être formulée.

ABSTRACT

Two different parallel lattice Boltzmann (LBM) algorithms have been devised for the simulation of flow through complex porous media. They are based on memory efficient LBM algorithms, namely the *one-lattice* and *shift* algorithms, combined with vector data structure, even fluid node vector partitioning domain decomposition and efficient data transfer layouts. The *shift* implementation also includes a single unit relaxation scheme that allows additional memory savings, but limits its validity to Newtonian fluids. They both provide high parallel performance by balancing the workload among the processors and reducing the amount of data that need to be transferred, and reduce significantly the memory usage as compared to previous parallel LBM codes presented in the literature. Theoretical parallel performance and memory usage models developed show that they also offer a good evolutivity and efficiencies as high as 79% for simulations made of several billions of fluid nodes on 128 processors are reported. The application of one of these algorithms for the simulation of flow through compressed packings made of highly polydisperse spheres has demonstrated the remarkable precision and efficiency of the algorithm proposed. As a result, a modified Carman-Kozeny correlation taking into account the compression level and the particle polydispersity has been formulated.

TABLE DES MATIÈRES

LISTE DES FIGURES

LISTE DES TABLEAUX

LISTE DES SIGLES ET ABRÉVIATIONS

Symbole	Description	Unité
∇P	Gradient de pression appliquée	Pa/m
δ	Tenseur d'identité	-
δ_t	Pas de temps	s
δ_x	Taille de maille du réseau	m
ΔP	Perte de charge	Pa
ΔQ	Variation de la quantité de mouvement	kg/m^3
ε	Porosité disponible pour l'écoulement	m/m
ε_i	Porosité disponible pour l'écoulement dans le sous-domaine i	m/m
ε_{max}	Porosité maximale des sous-domaines	m/m
μ	Viscosité newtonien du fluide	Pa·s
ν	Viscosité cinématique du fluide	m^2/s
ρ	Densité du fluide	kg/m^3
τ	Temps de relaxation	s
τ^*	Temps de relaxation adimensionnel	-
Ω	Opérateur de collision	$kg/(s \cdot m^3)$
Ω_i	Opérateur de collision dans la direction i	kg/m^3
A	Facteur d'accélération parallèle	-
$\mathcal{A}$	Terme d'advection de l'équation de Boltzmann	$kg/(s \cdot m^3)$
AGR	Automate de gaz sur réseau	-
c_K	Constante dite de Kozeny	-
Co_ν	Nombre de Courant diffusif ou visqueux	-
Co_u	Nombre de Courant convectif	-

c_s	Vitesse du son du réseau	m/s
$\mathcal{D}$	Nombre de dimensions de l'espace considéré	-
D2Q9	Réseau bidimensionnel à 9 directions	-
D3Q15	Réseau tridimensionnel à 15 directions	-
D3Q19	Réseau tridimensionnel à 19 directions	-
D3Q27	Réseau tridimensionnel à 27 directions	-
E	Efficacité parallèle	%
$\mathcal{E}$	Energie cinétique	J
$\mathbf{e}_i$	Direction de propagation dans la direction i	m/s
$\mathbf{f}$	Force externe ou *body force*	kg/m^3
F(i,j,k,b)	Structure de données matricielle	-
$f(\mathbf{x},\mathbf{e},t)$	Fonction de distribution de probabilité d'une particule	kg/m^3
$f^{eq}(\mathbf{x},\mathbf{e},t)$	Fonction de distribution d'équilibre locale	kg/m^3
$f_i(\mathbf{x},t)$	Fonction de distribution de probabilité d'une particule ou population dans la direction i	kg/m^3
$f_i^{eq}(\mathbf{x},t)$	Fonction de distribution d'équilibre locale ou population à l'équilibre dans la direction i	kg/m^3
f_s	Fraction des calculs	-
FSB	Front Side Bus	-
$\mathbf{g}$	Force gravitationnelle	m/s^2
HPC	High-Performance Computing	-
$\mathbf{K}$	Tenseur de perméabilité	m^2
k	Perméabilité	m^2
k_B	Constante de Boltzmann	J/K
Kn	Nombre de Knudsen	-
L	Distance sur laquelle s'applique la perte de charge ΔP	m
LBM	Lattice Bolztmannn method	-
m	Nombre d'opérations arithmétiques double précision effectué par nœud fluide par itération	-

Ma	Nombre de Mach	-
MBR	Méthode de Boltzmann sur réseau	-
N_A	Nombre d'Avogadro	mol^{-1}
n_b	Nombre de nœuds de rebond du réseau	-
n_c	Nombre de nœuds périodiques du réseau	-
n_d	Nombre de directions du réseau	-
n_f	Nombre de nœuds fluides du réseau	-
$n_i(\mathbf{x},t)$	Variable booléenne représentant la présence ou l'absence ou la présence d'une particule dans la direction i sur un automates de gaz sur réseau	-
n_{it}	Nombre d'itération	-
n_p	Nombre de processeurs	-
N_{tot}	Taille totale du domaine exprimée en nombre de nœuds du réseau	-
n_x, n_y, n_z	Dimensions du domaine en x, y et z respectivement	m
P	Pression	Pa
P_{com}	Performance parallèle des calculs en millions de mises à jour par seconde (MLUPS)	MLUPS
q_{com}	Nombre de réels double précision à échanger entre processeurs dans une direction	-
q_{mem}	Quantité de mémoire requise pour une simulation	octet
r_{cc}	Rapport entre communications et calculs	-
R	Constante des gaz parfaits	J/(K.mol)
RAM	Random Access Memory	-
Re	Nombre de Reynolds	-
s	Quantité d'octets à échanger entre processeurs dans une direction	octet
S_o	Surface spécifique de la phase solide	m^2/m^3
t	Temps	s

T	Température	°C
t_{cal}	Temps passé pour les calculs par itération	s
t_{com}	Temps passé pour les communications par itération	s
t_{CPU,n_p}	Temps d'exécution du programme parallèle sur n_p processeurs	s
t_{don}, t_{data}	Temps requis pour transmettre une donnée	s
t_i	Poids de la population i	-
t_{lat}	Latence	s
t_{oper}	Temps moyen passé pour une opération arithmétique	s
t_p	Temps d'exécution du programme parallèle	s
t_s	Temps d'exécution du programme séquentiel	s
u	Vitesse locale du fluide	m/s
U	Vitesse caractéristique (norme)	m/s
$\langle \mathbf{v} \rangle$	Vitesse moyenne superficielle du fluide	m/s
w_i	Poids de la population i	-
x	Vecteur de coordonnées	m

LISTE DES ANNEXES

CHAPITRE 1

INTRODUCTION

1.1 PROBLÉMATIQUE DE LA MÉCANIQUE DES FLUIDES NUMÉRIQUE EN MILIEU POREUX

Les milieux poreux sont sans contredit parmi les structures les plus complexes que la nature ait à offrir, comme peut en témoigner la reconstruction tomographique tridimensionnelle de l'espace poreux d'un morceau de grès de Fontainebleau[1] (Figure 1.1). La complexité de tels milieux réside avant tout dans la forte variation locale de la taille des pores et de leur connectivité. La connaissance des phénomènes de transport en milieux poreux, et plus précisément de l'écoulement de fluides au travers de milieux poreux, est non seulement d'un grand intérêt scientifique mais aussi d'une grande importance technologique : elle peut être appliquée à des domaines aussi variés que la

Figure 1.1 – Reconstruction microtomographique tridimensionnelle de l'espace poreux d'un morceau de grès de Fontainebleau [1].

production de l'énergie (extraction du gaz et du pétrole, piles à combustibles, géothermie), le génie chimique (réacteurs à lits à garnissage, moulage de polymères par injection), la fabrication du papier (couchage et impression du papier), le contrôle thermique des engins spatiaux (pompes capillaires) et la protection de l'environnement (dépollution des sols, stockage des déchets radioactifs). La complexité de ces écoulements tient au fait que les échanges de masse, de quantité de mouvement et d'énergie prennent place sur une large gamme d'échelles de longueurs et de vitesses.

La science de l'écoulement de fluides au travers de milieux poreux tire ses origines au milieu du XIX^{ème} siècle des travaux du célèbre ingénieur français Henri Darcy. Étudiant la percolation au travers d'entassements de grains de sable, il trouva que le débit de liquide s'écoulant au travers de ce milieu poreux est proportionnel à la force qui lui est appliqué. Cette découverte déboucha sur la célèbre équation connue sous le nom de loi de Darcy et valide seulement à faible nombre de Reynolds[2] [2]:

$$\langle \mathbf{v} \rangle = -\frac{\mathbf{K} \cdot [\nabla P - \rho \mathbf{g}]}{\mu} \tag{1.1}$$

où $\langle \mathbf{v} \rangle$ représente la vitesse moyenne superficielle du fluide, μ la viscosité du fluide, ∇P le gradient de pression appliqué, $\rho \mathbf{g}$ la densité de la force gravitationnelle et $\mathbf{K}$ le tenseur de perméabilité avec $\mathbf{K} = k\boldsymbol{\delta}$ pour des milieux homogènes isotropes, où $\boldsymbol{\delta}$ est le tenseur identité et k la perméabilité. La perméabilité est la mesure la plus couramment utilisée pour caractériser l'écoulement de fluides dans les milieux poreux. Elle correspond à la conductivité du milieu poreux pour l'écoulement d'un fluide donné et a pour dimensions des unités d'aire. C'est une propriété matérielle extrêmement sensible aux variations du milieu poreux. Par exemple, il n'est pas rare de voir la perméabilité k varier de plusieurs ordres de grandeur pour différentes feuilles de papier [3].

---

[1] Le grès de Fontainebleau est une roche couramment utilisée dans l'étude des milieux poreux du fait de sa pureté, de la relative uniformité des particules qui la composent (~ 200 μm) et de sa forte porosité (de 3 à 30%). En soi, il constitue un milieu très désordonné de forte complexité.

[2] C'est-à-dire en régime laminaire, condition très souvent satisfaite dans le cas de milieux poreux étant donné les faibles dimensions des pores qui leur sont en général associées.

Pour déterminer la perméabilité, il existe dans la littérature scientifique nombre de modèles semi-heuristiques ou empiriques basés sur des hypothèses d'ordre statistique, réseautique ou purement géométrique [2-8]. Le plus connu et utilisé d'entre eux est sans contredit le modèle de Carman-Kozeny[3] datant des années 1930 reliant la perméabilité à la porosité ε du milieu poreux et à la surface spécifique S_o des solides qui le composent[4] :

$$k = \frac{1}{c_K} \frac{1}{S_o^2} \frac{\varepsilon^3}{(1-\varepsilon)^2}, \tag{1.2}$$

où c_K est la constante dite de Kozeny reliée à la tortuosité. Celle-ci a été déterminée empiriquement pour des entassements de particules mono-disperses ($c_K \approx 5,0$). Faute de mieux, cette dernière valeur est en fait souvent utilisée pour toutes sortes d'autres milieux poreux et ce malgré de nombreuses données expérimentales montrant que pour des milieux hautement compressés, composés de particules polydisperses ou non-sphériques, la corrélation de Carman-Kozeny avec $c_K \approx 5,0$ dévie notablement des valeurs expérimentales et devrait être utilisée avec grande précaution [2]. Ces modèles empiriques restent donc très approximatifs. Comme il est expérimentalement très difficile de mesurer la perméabilité pour des milieux complexes avec la précision et le systématisme nécessaires, la seule façon théorique précise de calculer la perméabilité à partir de la connaissance microgéométrique du milieu reste l'intégration numérique des équations de Navier-Stokes (équations du mouvement). Cependant, dans le passé, les méthodes traditionnelles de mécanique des fluides numérique telles que les méthodes d'éléments finis, de différences finies, de volumes finis ou spectrales se sont avérées peu efficaces quant à la simulation des écoulements de fluides dans ce type de géométries complexes. Les principales raisons de l'échec des méthodes traditionnelles sont reliées à leur demande prohibitive en mémoire, à leurs parallélisations souvent ardues et/ou à la difficulté à générer des maillages adaptatifs adéquats pour des géométries complexes.

---

[3] Ce modèle empirique suppose que le milieu poreux peut être approximé par une conduite dont le diamètre est égal à 4 fois le rayon hydraulique défini comme le rapport entre le volume et la surface des pores.

Au début des années 90, une nouvelle approche de mécanique des fluides numérique a vu le jour : la méthode de Boltzmann sur réseau (MBR ou LBM pour *lattice Boltzmann method*, en anglais). Cette méthode n'a rien à voir dans ses fondements avec les méthodes précédemment utilisées, dans le sens où elle ne résout pas directement les équations différentielles partielles qui décrivent l'écoulement de fluides (équations de Navier-Stokes). MBR « simule » plutôt, par une approche particulaire, le comportement macroscopique de l'ensemble des molécules du fluide en mouvement. Pour être plus précis, un processus itératif de déplacements et de collisions de particules le long d'une grille structurée uniforme (aussi appelée réseau) discrétisant le domaine étudié permet d'obtenir en fonction des conditions frontières imposées un champ de densité de particules. À partir de celui-ci, les champs de vitesse et de pression du fluide peuvent ensuite être calculés. Bien que cette méthode n'ait pas encore atteint sa pleine maturité (tel que l'illustre la croissance exponentielle de la Figure 1.2) et que de nombreuses améliorations y soient régulièrement apportées, elle est déjà considérée par bon nombre comme la méthode par excellence pour la simulation des écoulements mono ou polyphasiques dans des géométries complexes telles que les milieux poreux [9-11]. Le nombre de publications et la complexité de certaines simulations utilisant MBR dans les dernières années est là pour en témoigner (voir Figure 1.3 et références [12-44]). Outre la

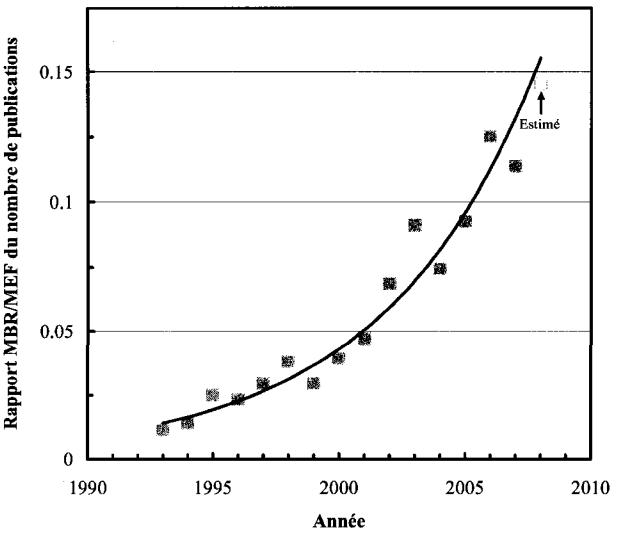


Figure 1.2 – Rapport du nombre de publications scientifiques entre MBR et la méthode des éléments finis (MEF) pour la période 1993-2008 tel que mesuré à partir de la base de donnée de l'Institut Canadien de l'Information Scientifique et Technique.

[4] La porosité étant définie comme le volume de pores accessible à l'écoulement par unité de volume du milieu poreux et la surface spécifique comme le rapport entre la surface et le volume des solides qui composent le milieu poreux.

simplicité d'implantation de MBR, sa supériorité par rapport aux méthodes conventionnelles découle de deux principaux facteurs :

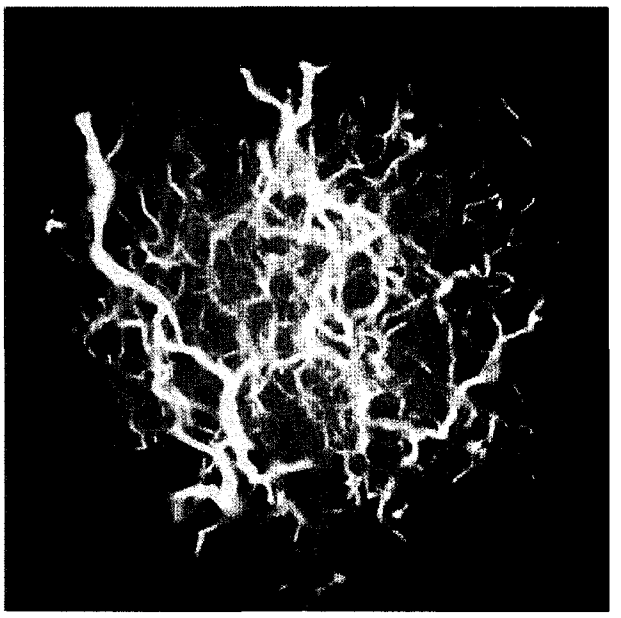


Figure 1.3 – Simulation MBR d'un écoulement d'un fluide (champ de vitesse) au travers d'un grès de Fontainebleau [32].

1. la souplesse à discrétiser le domaine d'étude au moyen d'une grille structurée uniforme de résolution suffisante sur laquelle les phases fluides et solides sont encodées de façon booléenne, les calculs ne prenant place que sur les nœuds représentant la phase fluide;
2. la facilité à la parallélisation qui provient de l'inhérente localité des règles de propagation et de collision qui sous-tendent la méthode. Avec l'avènement des ordinateurs parallèles permettant en théorie de diviser le temps de calcul d'un problème donné par le nombre de processeurs qui lui sont consacrés, le parallélisme d'un algorithme est devenu une propriété très recherchée.

Malgré tous ces avantages, trois avenues d'amélioration des performances de MBR ont attiré l'attention des chercheurs dans les dernières années et se doivent d'être considérées simultanément:

1. la réduction de l'utilisation de la mémoire par des simplifications algorithmiques, l'emploi de structures de données vectorielles ou l'usage de méthodes de maillage adaptatif ou « sans maillage» réduisant localement le nombre de nœuds de discrétisation ;
2. l'amélioration de la vitesse de calcul et de la précision de MBR grâce au développement de nouveaux algorithmes plus performants;

3. l'amélioration de l'efficacité du calcul parallèle à l'aide de techniques avancées de décomposition de domaine.

Le troisième point est particulièrement difficile à satisfaire pour ce qui est des milieux poreux ou à géométries complexes. Dans le cadre d'une décomposition de domaine, tout problème de charge de communication mis à part, une bonne efficacité parallèle requiert que les processeurs reçoivent des sous-domaines possédant des charges de calcul équivalentes, c'est-à-dire un bon équilibrage des tâches à exécuter. Sans cela, les processeurs les moins occupés passeront leur temps à attendre ceux qui le sont plus et l'efficacité du calcul parallèle s'en ressentira. Or, une simple partition cartésienne d'un domaine poreux hétérogène a toutes les chances de créer, par définition, des sous-domaines plus ou moins poreux. Étant donné que MBR n'exécute des calculs que sur les nœuds de la grille dits « fluides »[5], un sous-domaine de plus forte porosité aura nécessairement plus de calculs à exécuter qu'un autre de plus faible porosité. Toutefois, dans une perspective évolutive[6], même des milieux poreux apparemment homogènes (à moins d'être fractaux) peuvent créer une hétérogénéité dans la charge de calcul. Ceci est illustré par la Figure 1.4, où un grès de Fontainebleau apparemment homogène à grande échelle (porosité constant de 15,0% à ~4,7

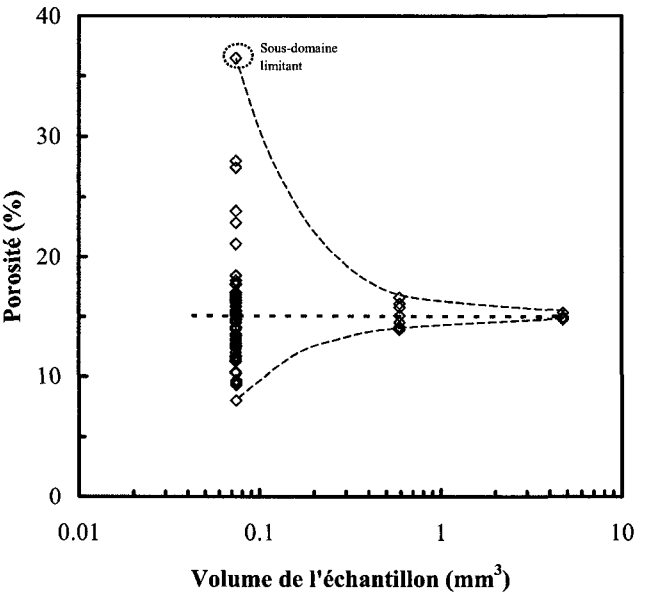


Figure 1.4 – Variations expérimentales de la porosité d'un grès de Fontainebleau en fonction du volume de l'échantillon observé (graphique obtenu à partir de données de microtomographie aux rayons X publiées dans [45]).

[5] Par opposition aux noeuds « solides » correspondant aux noeuds situés dans les parties solides du domaine.

[6] C'est-à-dire lorsque l'on accroît le nombre de sous-domaines pour un problème donné.

mm^3) peut apparaître très hétérogène à plus petite échelle (porosité variant entre 8,0 et 36,5% à ~0.08 mm^3). Une décomposition de domaine qui mènerait à ce dernier cas de figure serait extrêmement déséquilibrées et le temps de calcul en parallèle serait directement fonction du temps de calcul du domaine à 36,5% de porosité. L'homogénéité d'un milieu est donc dépendante de l'échelle à laquelle on le considère et, dans ces circonstances, l'évolutivité d'une méthode basée sur une décomposition cartésienne classique en sera grandement affectée. Des stratégies de parallélisation avancées et adaptées à MBR pour la simulation d'écoulement de fluides dans les milieux poreux se doivent d'être améliorées et combinées dans la mesure du possible avec les derniers développements algorithmiques de MBR. Par ailleurs, l'accroissement des performances et la grande diversité d'architectures[7] au niveau des ordinateurs parallèles n'aident à priori en rien le développement de stratégies de parallélisation « universelles ». La portabilité doit aussi être considéré lors du choix d'une méthode de parallélisation.

1.2 OBJECTIF GÉNÉRAL

En conséquence, nous nous proposons, dans le cadre de la simulation de l'écoulement isotherme monophasique d'un fluide Newtonien dans des milieux poreux saturés au moyen d'une méthode de Boltzmann sur réseau, de développer et d'étudier des algorithmes parallèles permettant d'améliorer la performance du calcul parallèle. Pour être pleinement efficaces, ces algorithmes se devront de tenir compte des dernières avancées algorithmiques en ce qui a trait à la réduction du temps calcul en séquentiel et de la quantité de mémoire utilisée. Le but ultime étant avant tout de simuler le plus rapidement possible l'écoulement dans des milieux poreux donnés les plus larges et complexes possibles (autrement dit en utilisant le moins de mémoire possible), il serait

[7] Que ce soit au niveau de l'accès à la mémoire, des réseaux de communication entre processeurs ou de l'hétérogénéité des processeurs.

sans doute peu judicieux de choisir une méthode que sur le seul critère de la performance parallèle. En autant que possible, une attention toute particulière sera apportée à l'évolutivité et la portabilité des stratégies parallèles. Ainsi, les stratégies seront principalement évaluées dans le cadre d'architectures MIMD (*Multiple Instruction Multiple Data* en anglais) à mémoire distribuée ou hybride reconnues comme étant d'une bien meilleure évolutivité que les architectures MIMD à mémoire partagée à cause de leurs problèmes d'engorgement reliés à l'accès à la mémoire. Finalement, les algorithmes ainsi développés seront testés sur leur capacité à résoudre des cas concrets permettant de faire avancer les connaissances scientifiques dans le domaine des écoulements de fluides en milieu poreux.

1.3 ORGANISATION DE LA THÈSE

Tout d'abord, étant donné les différentes implantations de MBR qui ont vu le jour récemment, nous ferons au Chapitre 2 une brève description historique de MBR et une revue des différentes implantations disponibles de façon à choisir la ou les plus efficaces. Par la suite, au Chapitre 3, nous aborderons les aspects parallèles de MBR et les différentes stratégies parallèles qui s'offrent à nous et définirons les objectifs spécifiques de la thèse. Dans les Chapitres 4 et 5, les différents stratégies parallèles développées dans le cadre de ces travaux ainsi que leurs performances respectives seront présentées au moyen de deux articles soumis à la revue scientifique *Computer & Fluids*. Le Chapitre 6 permettra de tester et de valider ces algorithmes sur des cas concrets d'écoulement dans des milieux poreux. Ceci se fera au moyen d'un troisième article paru dans *Computers & Chemical Engineering*. Finalement, nous discuterons les résultats obtenus au Chapitre 7 et conclurons au Chapitre 8 sur ceux-ci et sur leur portée, ainsi que sur les perspectives d'amélioration des algorithmes proposés et leurs futures applications.

CHAPITRE 2

MÉTHODE DE BOLTZMANN SUR RÉSEAU

Depuis sa création dans les années 90, plus de 1400 articles scientifiques ont été écrits sur le sujet. L'engouement autour de MBR est alimentée en partie par le développement quelque peu heuristique de la méthode : de nombreuses variantes et extensions ont vu et continuent de voir le jour régulièrement. Comme l'ont récemment souligné certains chercheurs [9], cet engouement a mené à une situation qui est souvent source de controverse et de confusion quant aux possibilités et applications de la méthode. Toutefois, MBR comme toute méthode possède ses contraintes et ses limitations. Au cours de ce chapitre, nous replacerons donc dans un premier temps le développement de MBR dans son contexte historique[8] avant de clarifier et dégager les grandes lignes de la méthode telles que présentées dans [9]. Nous terminerons ensuite par une revue des derniers développements dans le domaine des méthodes de Boltzmann et évaluerons celle qui présente le plus de potentiel quant à la simulation en parallèle des écoulements dans les milieux poreux.

2.1 ORIGINES

Le but n'est pas ici de faire un historique exhaustif des méthodes qui ont mené au développement de MBR (nous renvoyons pour cela le lecteur aux références [46-49]),

[8] Le lecteur pressé et non intéressé par les aspects historiques pourra directement passé à la Section 2.2.

mais il est bon toutefois d'en rappeler brièvement les grandes lignes pour mieux apprécier la méthode.

2.1.1 Automates cellulaires

Les origines de MBR remontent en fait à l'invention des automates cellulaires. À la fin des années 40, aux prémices de l'aire informatique, les célèbres mathématiciens Stanislaw Ulam et John von Neumann proposent de programmer les ordinateurs pour « simuler la vie ». Leur idée est de diviser tout d'abord l'espace du problème au moyen de cellules ou nœuds formant un réseau structuré, chacun des nœuds étant capable de représenter un nombre fini d'états et l'état d'un nœud à une « génération » donnée étant affecté par l'état des nœuds voisins lors de la génération précédente selon certains règles récursives. Ainsi, l'état des nœuds évolue de génération en génération, en général, d'un état initial vers une situation d'équilibre. Ulam et von Neumann suggérèrent que sous certaines conditions (c'est-à-dire pour des règles récursives bien déterminées qui sont en général trouvées par expérimentation), le comportement collectif du système tout entier peut être suffisamment complexe qu'il tend à imiter certains aspects de systèmes biologiques réels, comme par exemple la reproduction d'organismes vivants. Ainsi, en 1967, Convay proposa son célèbre « Game of Life » (cf. Figure 2.1) [50]. Il consiste en une grille bidimensionnelle théoriquement infinie dont chacune des cellules peut contenir un « organisme » à la fois et possède 8 cellules

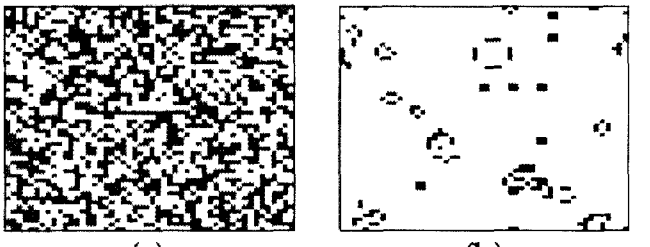


Figure 2.1 – « Game of Life » pour un domaine de 50 par 50 cellules avec conditions périodiques. Initialisation aléatoire (a) d'égale probabilité d'organismes vivants (gris) et morts (blanc) et populations à l'équilibre (b) [48].

voisines, incluant les cellules diagonalement adjacentes. À partir d'un état initial, les règles suivantes sont appliquées de génération en génération :

1. Chaque organisme avec 2 ou 3 organismes voisins survit jusqu'à la prochaine génération;
2. Chaque organisme avec 4 organismes voisins ou plus meurt à cause de surpopulation;
3. Chaque organisme avec un organisme voisin ou moins meurt d'isolation;
4. Chaque cellule vide adjacente à exactement 3 cellules occupées donne naissance à un organisme.

L'établissement de ces règles simples permet ainsi d'imiter la croissance microbienne, mais furent dérivées après de longues périodes d'expérimentation. Comme autres automates cellulaires du même genre, citons aussi le « Sharks and Fishes » et le « Foxes and Rabbits » permettant d'étudier l'évolution des populations de certains animaux en fonction de leur prédateurs, des ressources disponibles et des conditions initiales des populations [51]. Les automates cellulaires s'avèrent en fait d'une grande utilité dans de nombreux domaines, comme par exemple dans la gestion du trafic autoroutier où le comportement individuel des automobilistes peut être facilement encodé [52].

2.1.2 Automates de gaz sur réseau

En leur temps, Ulam et von Neumann proposèrent aussi d'utiliser cette méthode pour résoudre des problèmes de physique et de mathématiques. L'idée après tout est séduisante : s'affranchir de trouver et de résoudre les équations différentielles partielles qui gouvernent le comportement d'un système donné en simulant par des règles

« simples » le comportement du dit système[9]. Il fallut toutefois attendre les travaux de Frisch, Hasslacher et Pomeau en 1986 [53] pour finalement obtenir des règles qui permettent de décrire la mécanique des fluides. Les automates de gaz sur réseau (AGR ou *lattice-gas automata*, en anglais) étaient nés et de ceux-ci découlait une découverte saisissante : le comportement d'un fluide dépend très peu de la nature des particules qui constituent ce fluide à partir du moment où ce fluide est examiné à une échelle bien supérieure à celle des particules ($\mathrm{Kn} < 0{,}01$)[10]. Ainsi, pour simuler par exemple l'écoulement de l'eau, rien ne sert de prendre des particules qui ont la taille et la forme des molécules d'eau. Nous pouvons tout simplement prendre des particules sphériques virtuelles dont la taille surpasse de plusieurs ordres de grandeur celle des molécules d'eau à la condition que la règle énoncée ci-haut tienne toujours. L'automate bidimensionnel de gaz sur réseaux inventé par Frisch, Hasslacher et Pomeau (que nous nommerons dorénavant FHP pour des raisons de concision) repose donc sur ce principe et sur

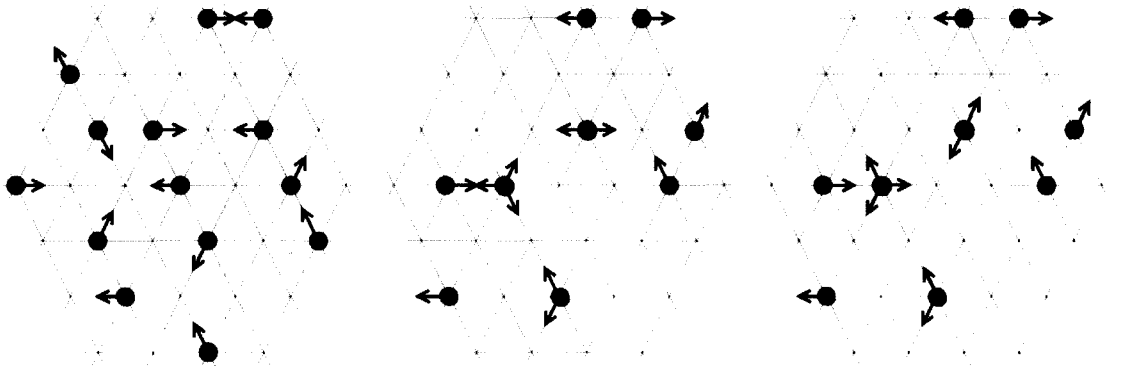


Figure 2.2 – Phases de propagation (b) et de collision (c) pour FHP à partir d'un état initial (a). Les flèches dénotent les particules virtuelles et leurs directions respectives [49].

[9] Lors d'une conférence en 1950, Ulam posa le problème en ces termes [46]: « Given a partial differential equation, we construct models of suitable games, and obtain distributions or solutions of the corresponding equations by playing these games, i.e., by experiment ».

[10] On définit pour cela le nombre de Knudsen : $\mathrm{Kn} = l_{lpm}/L_{hydro}$ où L_{hydro} est la plus courte longueur caractéristique de l'écoulement observé (par exemple, la taille de maille du réseau) et l_{lpm} le libre parcours moyen des molécules entre deux collisions. Il est généralement admis que l'écoulement du fluide peut être considéré comme continu lorsque $\mathrm{Kn} < 0{,}01$. Lorsque $0{,}01 < \mathrm{Kn} < 0{,}10$ le régime d'écoulement est dit « glissant » et transitoire pour $0{,}1 < \mathrm{Kn} < 3$.

quelques règles simples de déplacement des particules virtuelles sur une grille hexagonale. Ces règles peuvent être subdivisées en deux étapes principales :

1. une phase de propagation synchrone des particules le long des mailles de la grille à partir d'un état donné du système (Figure 2.2(b));

2. une phase de collision soumise à une contrainte de conservation de la masse et de la quantité de mouvement et servant à gérer l'interaction des particules concourantes (Figure 2.2(c) et Figure 2.3).

En termes mathématiques, le modèle FHP peut s'écrire comme suit :

$$n_i(\mathbf{x}+\mathbf{e}_i \times 1, t+1) = n_i(\mathbf{x}, t) + \Omega_i\left(\mathbf{n}(\mathbf{x}, t)\right) \tag{2.1}$$

où $n_i(\mathbf{x}, t)$ est la variable booléenne représentant la présence ou l'absence d'une particule animée d'une vitesse $\mathbf{e_i}$ (égale à l'unité pour FHP) au nœud $\mathbf{x}$ et au temps t, et $\Omega_i\left(\mathbf{n}(\mathbf{x}, t)\right)$

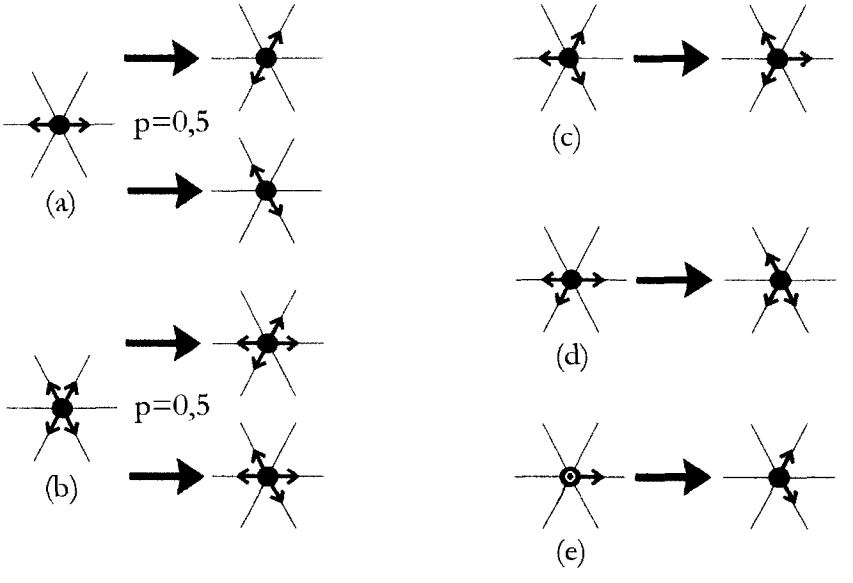


Figure 2.3 – Règles de collision pour FHP : collisions binaire (a), quaternaire (b), ternaires (c) et (d), et binaire avec une particule au repos (e). Lorsque deux choix de collisions sont possibles (a et b), une sélection aléatoire est faite.

est l'état de sortie de l'opérateur de collision dans la direction i connaissant l'état d'occupation au nœud **x** et au temps t dans toutes les directions. Les vitesses $\mathbf{e}_i$ sont définies dans le cas d'un réseau hexagonal comme suit :

$$\mathbf{e}_i = \left(\cos\left(\pi i/3\right), \sin\left(\pi i/3\right)\right), \quad i = 1,2,\ldots,6. \tag{2.2}$$

L'opérateur de collision dans le cas d'un réseau FHP à 7 vitesses (c'est-à-dire les 6 vitesses plus une vitesse nulle pour les particules au repos[11]) est présenté à la Figure 2.3. Il a été démontré [46-49] que ces simples règles suffisent pour retrouver le comportement des équations de Navier-Stokes pour un fluide incompressible à la condition que le nombre de Mach[12] soit faible. Il est aussi à noter que le choix du type de réseau (c'est-à-dire de la grille hexagonale pour FHP) n'est pas anodin : en plus de pouvoir remplir l'espace du domaine adéquatement, celui-ci doit posséder certaines règles de symétrie de façon à obtenir un comportement hydrodynamique isotrope et éviter ainsi l'obtention de lois de conservation aphysiques[13]. Malheureusement, il n'existe aucun polyèdre en 3D qui satisfasse ces contraintes. En conséquence, pour étendre AGR au cas tridimensionnel, un réseau basé

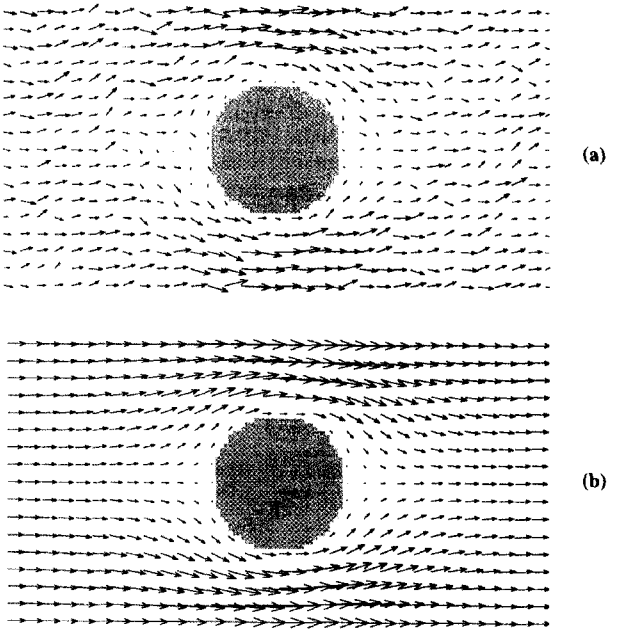


Figure 2.4 – Comparaison entre AGR (a) et MBR (b) pour l'écoulement autour d'un cylindre [46].

[11] Les particules au repos ne sont pas obligatoires mais sont ajoutées de manière à accroître le taux de collision, permettant ainsi au système de relaxer plus rapidement vers l'équilibre statistique, réduisant la viscosité et ainsi améliorant l'invariance galiléenne [46,49].

[12] Le nombre de Mach est le rapport de la vitesse caractéristique (U) de l'écoulement sur la vitesse du son dans le fluide considéré (c_s), c'est-à-dire le rapport entre des forces d'inertie et le gradient de pression.

[13] Ce point retarda en fait l'invention de FHP de près d'une quinzaine d'année car une version préliminaire de FHP proposée par Hardy, de Pazzis et Pomeau [46] et basée sur une grille orthogonale avait démontré un comportement anisotrope, ce qui avait eu pour effet de décourager bon nombre de chercheurs.

sur la projection d'un hypercube à faces centrées possédant les propriétés isométriques requises en 4 dimensions fut développé [48].

Outre cette apparente complexité des réseaux à trois dimensions, l'inconvénient majeur de AGR provient des fluctuations statistiques sur les solutions obtenues (cf. Figure 2.4). Celles-ci découlent directement de la nature particulaire booléenne qui sous-tend la méthode. En fait, pour obtenir des champs d'écoulement précis, un moyennage sur un grande nombre de pas de temps est souvent nécessaire. Pour répondre à ce problème, McNamara et Zanetti [54] proposèrent en 1988 d'utiliser en lieu et place de particules virtuelles des populations ou distributions de probabilité de particules par le truchement de l'équation de Boltzmann. La méthode de Boltzmann sur réseau voyait le jour et allait rapidement se détourner des automates de gaz sur réseaux, ne retenant principalement de ceux-ci que leurs avantages et résolvant nombre de leurs inconvénients[14].

2.2 PRINCIPE DE LA MÉTHODE

Nous nous garderons bien ici de rentrer dans les détails des différentes variantes de MBR qui sont apparues au cours des ans et qui découlent du développement quelque peu heuristique de la méthode. Nous nous bornerons donc à décrire les principes fondateurs de la méthode la plus généralement citée dans la littérature, tout en soulignant ses contraintes et ses limitations. Encore une fois, pour de plus amples détails, nous référons le lecteur aux excellentes références suivantes [9,46-47].

[14] En fait, le seul avantage encore concédé réside dans les erreurs d'arrondis absents dans AGR grâce à la nature booléenne de la méthode.

2.2.1 Théorie cinétique des gaz et équation de Boltzmann

L'équation de Boltzmann repose sur la théorie cinétique des gaz et décrit l'évolution de la fonction de distribution de probabilité d'une particule $f(\mathbf{x},\mathbf{e},t)$ (ou population) en fonction de ses interactions microdynamiques[15] :

$$\left(\partial_t + \mathbf{e}\cdot\nabla_x + \mathbf{g}\cdot\nabla_e\right) f\left(\mathbf{x},\mathbf{e},t\right) = \Omega\left(f\left(\mathbf{x},\mathbf{e},t\right)\right) \tag{2.3}$$

où $\mathbf{x}$ sont les coordonnées de la particule, $\mathbf{e}$ sa vitesse, t le temps, $\mathbf{g}$ une force externe s'appliquant sur la particule et où Ω représente un terme de collision binaire interparticulaire supposant que la vitesse de la particule est non corrélée avec sa position[16]. Le plus difficile dans l'Équation (2.3) est de déterminer le terme de collision. En 1954, Bhatnager, Gross et Krook proposèrent d'approximer le terme de collision au moyen d'un processus de relaxation vers la fonction de distribution d'équilibre locale $f^{eq}(\mathbf{x},\mathbf{e},t)$ [55] :

$$\Omega\left(f\left(\mathbf{x},\mathbf{e},t\right)\right) = -\frac{f\left(\mathbf{x},\mathbf{e},t\right) - f^{eq}\left(\mathbf{x},\mathbf{e},t\right)}{\tau} \tag{2.4}$$

où τ est le temps de relaxation unique[17]. Cette approximation « phénoménologique » est devenue avec le temps un des fondements de MBR, à tel point que les réseaux utilisant cette approximation sont souvent appelés réseaux BGK ou MBR-BGK. La principale hypothèse de BGK est que le fluide est localement proche de l'équilibre thermique. L'idée est que les collisions poussent le système vers cet état d'équilibre local défini d'après la théorie des intervalles de collision. Celle-ci stipule que durant un intervalle de

[15] $f(\mathbf{x},\mathbf{e},t)$ représente la fonction de distribution de probabilité d'une particule localisée au point $\mathbf{x}$ au temps t et animée d'une vitesse $\mathbf{e}$. $[f(\mathbf{x},\mathbf{e},t)\cdot dx^3\cdot de^3]$ est donc le nombre de particules qui, au temps t, sont localisées dans l'élément de contrôle spatial $[dx^3\cdot de^3]$.

[16] C'est ce que l'on appelle l'hypothèse du chaos moléculaire.

[17] Il existe d'autres modèles avec des temps de relaxation multiples.

temps δ_t une fraction δ_t/τ des particules dans un petit volume donné subit des collisions qui altère f vers un équilibre défini par la distribution de Maxwell-Boltzmann suivante :

$$f^{eq}(\mathbf{x},\mathbf{e},t)=\frac{\rho}{(2\pi RT)^{\mathcal{D}/2}}\exp\left[-\frac{(\mathbf{e}-\mathbf{u})^2}{2RT}\right] \tag{2.5}$$

où $\mathcal{D}$, R, T, ρ et $\mathbf{u}$ sont respectivement la dimension de l'espace considéré, la constante des gaz parfaits, la température, la densité et la vitesse du fluide. Dans le cas où la température est constante et les vitesses faibles (faible nombre de Mach), l'Équation (2.5) peut être approximée par un développement du deuxième ordre en vitesse :

$$f^{eq}(\mathbf{x},\mathbf{e},t)=\frac{\rho\exp\left[-\frac{\mathbf{e}^2}{2RT}\right]}{(2\pi RT)^{\mathcal{D}/2}}\times\left(1+\frac{(\mathbf{e}\cdot\mathbf{u})}{RT}+\frac{(\mathbf{e}\cdot\mathbf{u})^2}{2(RT)^2}-\frac{\mathbf{u}^2}{2RT}\right)+\mathcal{O}(\mathbf{u}^3) \tag{2.6}$$

Combinons maintenant les Équations (2.3), (2.4) et (2.5) en utilisant l'approximation $\nabla_e f(\mathbf{x},\mathbf{e},t)\approx\nabla_e f^{eq}(\mathbf{x},\mathbf{e},t)$. Nous obtenons l'équation suivante :

$$\partial_t f+\mathbf{e}\cdot\nabla_x f=-\frac{f-f^{eq}}{\tau}+\frac{\mathbf{g}\cdot(\mathbf{e}-\mathbf{u})}{RT}f^{eq} \tag{2.7}$$

où dorénavant f et $f(\mathbf{x},\mathbf{e},t)$ seront interchangeables.

Le lien entre l'équation de Boltzmann (Équation (2.3)) et l'hydrodynamique est effectué à travers l'intégration de f dans l'espace des vitesses de particules :

$$\rho=\int f\,d\mathbf{e},\quad \rho\mathbf{u}=\int(f\mathbf{e})\,d\mathbf{e},\quad \rho\mathcal{E}=\tfrac{1}{2}\int\left(f(\mathbf{e}-\mathbf{u})^2\right)d\mathbf{e} \tag{2.8}$$

avec l'énergie cinétique $\mathcal{E}$ définie comme suit :

$$\mathcal{E}=\frac{\mathcal{D}}{2}k_BT=\frac{\mathcal{D}}{2}\frac{RT}{N_A} \tag{2.9}$$

où k_B est la constante de Boltzmann[18] et N_A le nombre d'Avogadro.

2.2.2 Discrétisation de l'équation de Boltzmann

Il existe principalement deux façons de dériver les équations qui sous-tendent la méthode de Boltzmann sur réseau : une façon heuristique historique et une façon plus récente basée sur une approximation de type différence finie [9,56-57]. Nous nous contenterons de décrire la première étant donné qu'elle découle de AGR.

Pour arriver à la formulation de MBR, il convient tout d'abord de discrétiser dans l'espace des vitesses (ou des directions) l'équation de Boltzmann. Explicitement, nous faisons les changements de variables suivants :

$$\begin{aligned} f(\mathbf{x},\mathbf{e},t) &\longrightarrow f_i(\mathbf{x},t) \\ \mathbf{e} &\longrightarrow \mathbf{e}_i \\ f^{eq}(\mathbf{x},\mathbf{e},t) &\longrightarrow f_i^{eq}(\mathbf{x},t) \end{aligned} \tag{2.10}$$

où f_i, $\mathbf{e}_i$ et f_i^{eq} sont les variables équivalentes mais cette fois-ci exprimées pour une direction i correspondant à une des directions du réseau sous-jacent. Comme pour AGR, l'espace considéré est bien entendu préalablement discrétisé au moyen d'une grille ou réseau (*lattice* en anglais) respectant certains critères de symétrie. Toutefois, ces critères de symétrie sont relaxés par rapport à AGR par le fait que plusieurs grandeurs de vecteurs vitesses peuvent se côtoyer sur une même grille, ce qui n'est pas le cas avec AGR où tous les vecteurs vitesse se doivent d'avoir la même grandeur de façon à synchroniser les collisions des particules individuelles. Plusieurs réseaux polygonaux et polyédriques garantissent ainsi les lois de conservation. Parmi les plus couramment utilisés, mentionnons les réseaux D2Q9 formé de carrés et D3Q15 formé de cubes, où le chiffre

[18] Encore lui! Et dire qu'il s'est suicidé parce que personne ne croyait à sa théorie cinétique des gaz!

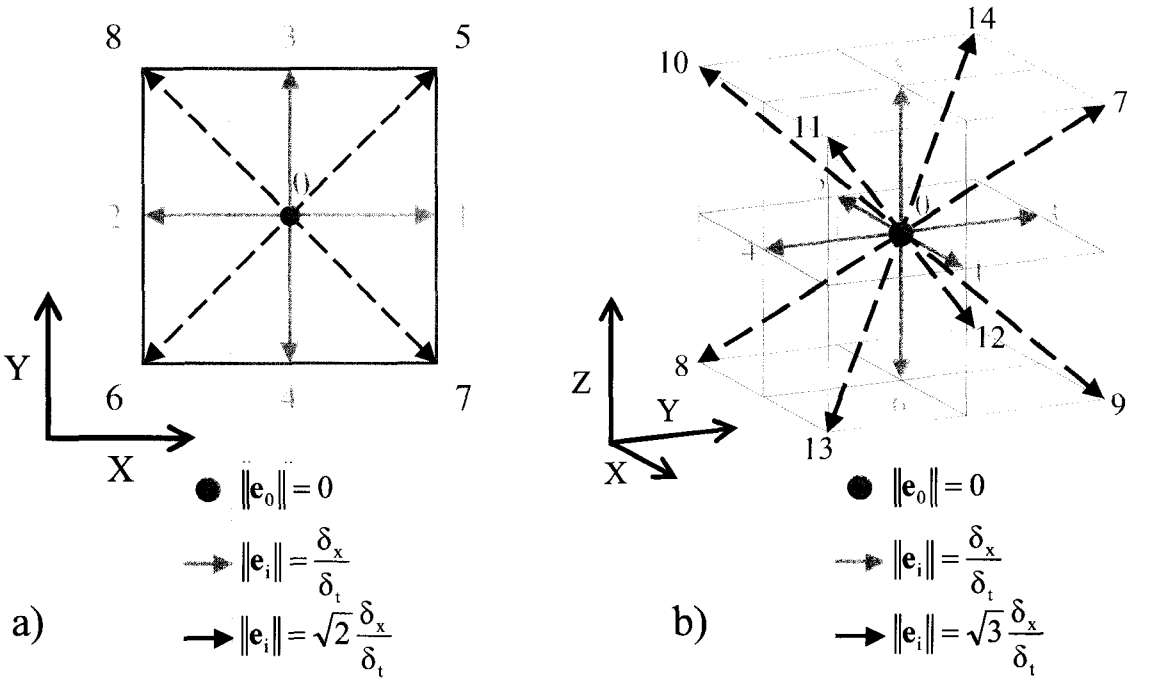


Figure 2.5 – Représentation des grilles carré D2Q9 (a) et cubique D3Q15 (b) et de leurs vecteurs vitesses respectifs.

après le D correspond à la dimensionnalité de la grille et le chiffre après le Q réfère au nombre de vitesses i disponibles (Figure 2.5 et Annexe A).

À partir de l'Équation (2.7) et des changements de variables proposés précédemment, on obtient l'équation de Boltzmann BGK discrète dans un repère Lagrangien :

$$\underbrace{\partial_t f_i + e_j \partial_j f_i}_{\text{Advection } A(f_i)} = \underbrace{-\frac{f_i - f_i^{eq}}{\tau} + \frac{g_j(e_j - u_j)}{c_s^2} f_i^{eq}}_{\text{Collision } \Omega(f_i)} \tag{2.11}$$

où g_j, et u_j correspondent respectivement aux projections de **g** et **u** dans la direction j, c_s est la vitesse du son du réseau définie par $c_s = \sqrt{RT} = \beta\frac{\delta_x}{\delta_t}$ et où β est une constante fonction du type de réseau utilisé (voir Annexe B). Le terme de collision englobe ici le

terme associé à la force externe. L'Équation (2.11) peut être résolue par divers algorithmes numériques. Nous examinerons ce point dans la prochaine sous-section.

Par analogie avec l'Équation (2.6), nous pouvons aussi écrire :

$$f_i^{eq}(\mathbf{x},t) = A_i + B_i(\mathbf{e}_i \cdot \mathbf{u}) + C_i(\mathbf{e}_i \cdot \mathbf{u})^2 + D_i\mathbf{u}^2 \tag{2.12}$$

La détermination des coefficients A_i, B_i, C_i et D_i a longtemps fait l'objet d'intenses développements plus ou moins heuristiques et ces coefficients étaient en fait à l'origine « ajustés » de façon à retrouver le comportement des équations de Navier-Stokes au moyen d'un développement perturbatif multi-échelle dit de Chapman-Enskog. Il en résulte souvent dans la littérature un flou relatif quant aux valeurs exactes que doivent prendre ces coefficients et les contraintes qui y sont associées. L'Annexe B donne ces coefficients tels que définis dans [9].

Finalement, les Équations (2.8) nous mènent à écrire les relations discrètes suivantes pour la masse volumique (densité) du fluide, sa quantité de mouvement et son énergie cinétique :

$$\rho = \sum_i f_i, \quad \rho\mathbf{u} = \sum_i f_i\mathbf{e}_i, \quad \rho\mathcal{E} = \tfrac{1}{2}\sum_i f_i(\mathbf{e}_i - \mathbf{u})^2 \tag{2.13}$$

2.3 IMPLANTATION NUMÉRIQUE DE MBR-BGK

Le schéma numérique le plus couramment utilisé pour résoudre l'Équation (2.11) est le schéma dit de « collision et propagation » (C&P). Étant donné sa simplicité et son large aura parmi la communauté scientifique, nous nous bornerons ici à décrire ce schéma tout en soulignant ses éventuelles faiblesses et en quoi les schémas plus avancés peuvent y remédier.

2.3.1 Schéma « collision et propagation »

Le schéma C&P implique une discrétisation en temps et en espace de l'équation de Boltzmann BGK discrète. La discrétisation du terme d'advection $\mathcal{A}(f_i)$ est réalisée au moyen d'un schéma de dérivation avant implicite précis au premier ordre dans un repère Lagrangien :

$$\begin{aligned}\mathcal{A}(f_i) &= \frac{f_i(\mathbf{x},t+\delta_t)-f_i(\mathbf{x},t)}{\delta_t}+\frac{\delta_j}{\delta_t}\frac{f_i(\mathbf{x}+\mathbf{e}_i\delta_t,t+\delta_t)-f_i(\mathbf{x},t+\delta_t)}{\delta_j} \\ &= \frac{f_i(\mathbf{x}+\mathbf{e}_i\delta_t,t+\delta_t)-f_i(\mathbf{x},t)}{\delta_t}\end{aligned} \tag{2.14}$$

alors qu'une discrétisation explicite précise au première ordre est appliquée à l'opérateur de collision. L'Équation (2.11) devient donc :

$$f_i(\mathbf{x}+\mathbf{e}_i\delta_t,t+\delta_t)-f_i(\mathbf{x},t)=-\frac{f_i(\mathbf{x},t)-f_i^{eq}(\mathbf{x},t)}{\tau^*}+\frac{g_i(e_i-u_i)\delta_t}{c_s^2}f_i^{eq}(\mathbf{x},t) \tag{2.15}$$

où δ_x et δ_t représentent les discrétisations en espace et en temps du réseau et $\tau^*=\tau/\delta_t$ est le temps de relaxation adimensionnel. On a aussi $\delta_i=\|\mathbf{e}_i\|\delta_t$, ce qui signifie que la population i se retrouve à l'itération suivante sur le nœud adjacent dans la direction i de celui qu'elle occupe actuellement (la distance séparant les nœuds étant de δ_i). Plus simplement exprimé, les populations se propagent de nœud en nœud à chaque itération. Notons de plus que le deuxième terme du membre de droite de l'Équation (2.15) est souvent supprimé (c'est-à-dire $\mathbf{g}=0$). En effet, comme nous le verrons plus tard, il existe d'autres moyens d'imposer une force. L'Équation (2.15) ainsi simplifiée mène donc à la formulation explicite de la méthode standard de MBR-BGK :

$$f_i(\mathbf{x}+\mathbf{e}_i\delta_t, t+\delta_t) = f_i(\mathbf{x},t) - \frac{f_i(\mathbf{x},t) - f_i^{eq}(\mathbf{x},t)}{\tau^*} = f_i(\mathbf{x},t) + \Omega_i(\mathbf{x},t) \tag{2.16}$$

Étant donné les discrétisations du premier ordre des opérateurs d'advection et de collision, on pourrait s'attendre à ce que le schéma C&P soit lui-même précis au premier ordre. Toutefois, comme souligné et démontré par [47], le schéma C&P possède en fait une précision du deuxième ordre en temps et en espace. Cela vient du fait que chaque population f_i est complétée par sa population conjuguée miroir transformant ainsi dans les faits le schéma de discrétisation en une différentiation finie centrée[19].

Quelques remarques s'imposent maintenant quant aux limitations et à la stabilité des algorithmes C&P. Rappelons tout d'abord que la méthode de MBR-BGK repose sur des hypothèses de faibles nombres de Mach et de Knudsen (Ma<<1 et Kn<<0,01). De plus, étant donné le schéma de différentiation explicite de l'algorithme, les conditions habituelles de Courant-Friedrichs-Lewy (CFL) doivent être vérifiées de façon à garantir la stabilité de la méthode. Nous invitons à ce stade le lecteur intéressé à se rapporter à l'Annexe D pour une discussion plus en détails de la stabilité de l'algorithme C&P.

Finalement, mentionnons que plusieurs autres algorithmes de précision et de stabilité supérieures à C&P ont été développés [9,58-59]. Il repose sur des discrétisations des opérateurs de collision et de propagation d'ordre supérieur. Toutefois, ces gains sont quelque peu ternis par des processus itératifs additionnels et donc un accroissement significatif des calculs requis par pas de temps, mais la localité des opérations reste cependant conservée.

[19] La précision du deuxième ordre est valide pour le cœur du domaine de calcul mais, en fonction des conditions frontières choisies, elle peut dans les faits se dégrader vers le premier ordre. On veillera donc à choisir des conditions frontières précises au deuxième ordre.

2.3.2 Évolution temporelle du schéma C&P et conditions frontières

Pour obtenir l'évolution des champs de vitesse et de densité dans le temps (calculés au moyen des Équations (2.13)), il suffit donc de faire évoluer l'Équation (2.16) par un processus itératif jusqu'à la convergence souhaitée (par exemple, $(\mathbf{du}/dt)/(\mathbf{du}/dt)_{max}=10^{-5}$) à partir de conditions frontières et initiales (par exemple, $\mathbf{u}=0$). En ce qui concerne les conditions frontières, il en existe une large variété de précisions diverses. Nous utiliserons ici les plus populaires, à savoir le rebond à mi-chemin (*half-way bounce-back* en anglais) pour des conditions de mur sans glissement et des conditions périodiques pour les frontières extérieures du domaine. La perte de charge est elle imposée au moyen d'une force externe (*body force* en anglais) appliquée sur tous les nœuds fluides du domaine dans la direction du gradient de pression. Le lecteur est invité à lire l'Annexe C pour une description plus détaillée de ces conditions frontières.

En ce qui a trait au choix de τ^*, celui-ci se fait par l'entremise de l'équation suivante reliant le temps de relaxation à la viscosité cinématique ν du fluide simulé:

$$\nu=\left(\tau^*-\tfrac{1}{2}\right)\delta_t c_s^2=\mu/\rho \qquad \Rightarrow \tau^*=\frac{\nu}{\delta_t c_s^2}+\tfrac{1}{2}=\frac{\nu\,\delta_t}{\beta\,\delta_x^2}+\tfrac{1}{2} \tag{2.17}$$

où μ est la viscosité newtonienne du fluide, ρ est, comme nous l'avons déjà mentionné, la densité du fluide[20,21]. Comme la viscosité est en général fixée, l'Équation (2.17) possède donc deux degrés de liberté parmi les trois paramètres δ_t, δ_x et τ^*. En théorie, il est très intéressant de fixer δ_x et de prendre τ^* très grand ($>\frac{1}{2}$) de façon à avoir le pas de temps δ_t le plus grand possible et ainsi converger plus rapidement vers la solution désirée. En

[20] Attention, dorénavant, nous utiliserons la notation $\rho(\mathbf{x},t)$ pour densité locale du fluide et ρ pour densité « moyenne » du fluide.

[21] Le facteur ½ dans l'Équation (2.17) est spécifique à l'algorithme C&P et est rajouté au terme de viscosité pour compenser l'erreur de discrétisation faite et ainsi réduire la diffusion numérique. Il est ainsi nécessaire pour que le développement multi-échelle de Chapman-Enskog de l'algorithme C&P récupère les équations de Navier-Stokes avec une précision du deuxième ordre [60].

pratique, la précision et la stabilité de l'algorithme peut dépendre du choix de τ^* en fonction des conditions frontières utilisées (c'est ce que nous verrons dans l'Article No.2 du Chapitre 5).

2.3.3 Algorithmes numériques classiques: *two-lattice* vs *one-lattice*

Il existe deux façons principales d'implanter le schéma général C&P: en une ou en deux étapes. Le premier, appelé aussi schéma fusionné, applique directement l'Équation (2.16) reformulée comme suit:

$$f_i(\mathbf{x},t) = f_i(\mathbf{x}-\mathbf{e}_i\delta_t, t-\delta_t) - \frac{f_i(\mathbf{x}-\mathbf{e}_i\delta_t, t-\delta_t) - f_i^{eq}(\mathbf{x}-\mathbf{e}_i\delta_t, t-\delta_t)}{\tau^*} \tag{2.18}$$

en une seule étape exécutée séquentiellement sur tous les nœuds fluides dans toutes les directions i. Historiquement, le schéma à une étape fut le premier à être implanté du fait de sa simplicité. Toutefois, étant explicite par nature, il requiert le stockage en mémoire des populations de deux itérations consécutives dans deux matrices tridimensionnelles distinctes (en 3D). En effet, le calcul des populations à un nœud donné au temps t requiert les populations de tous les nœuds environnants à ce nœud à l'itération t-δ_t. Or, l'utilisation d'une seule matrice et d'un balayage de l'ensemble des nœuds dans un sens donné, quel que soit ce sens a priori, reviendrait à écraser des populations à l'itération t-δ_t qui seraient éventuellement encore nécessaires pour les calculs de nœuds subséquents. Ce schéma à deux réseaux et une étape mène à l'algorithme dit *two-lattice (one-step) algorithm* d'après la nomenclature anglophone récemment proposée par Mattila *et al.* [61][22].

Pour réduire l'utilisation mémoire, l'algorithme à deux étapes n'utilisant qu'une matrice de stockage, soit le *one-lattice (two-step) algorithm*, a été mise au point. Les

[22] Nous adopterons en grande partie pour le reste de cette présentation la nomenclature anglophone proposée de façon à faciliter la lecture des articles des Chapitres 4 & 5 rédigés en anglais.

phases de collision et de propagation sont ici découplées: les collisions sont exécutées respectivement sur tous les nœuds dans toutes les directions i:

$$f_i^*(\mathbf{x},t) = f_i(\mathbf{x},t) - \frac{f_i(\mathbf{x},t) - f_i^{eq}(\mathbf{x},t)}{\tau^*} \tag{2.19a}$$

où $f_i^*(\mathbf{x},t)$ est une population post-collision et « prête à la propagation », et ensuite les propagations procèdent pour chacune des directions i dans un ordre de mise à jour de nœuds bien précis:

$$f_i(\mathbf{x}+\mathbf{e}_i\delta_t, t+\delta_t) = f_i^*(\mathbf{x},t) \tag{2.19b}$$

L'ordre de mise à jour des nœuds est effectué de telle manière que les populations post-collision sont propagées en ordre de **x** ascendant lorsque $\mathbf{e}_i$ pointe dans la direction -**x** et vice-versa. Ceci évite d'écrire par dessus une population post-collision non encore propagée. Pour que le tout marche, il est pratique d'avoir recours à deux catégories additionnelles de nœuds: 1) des nœuds dits de « rebond » (ou *bounce-back nodes* en anglais) et 2) des nœuds dits « périodiques » (ou *periodic nodes* en anglais). Les nœuds de rebond sont des nœuds solides directement connectés à un ou des nœuds fluides. Comme leur nom l'indique, on y copie et retourne les populations qui sont nécessaires pour exécuter le rebond à mi-chemin (condition de mur) avant la prochaine propagation. Les nœuds périodiques constituent un halo de nœuds tout autour du domaine qui sert à copier les populations entrantes du domaine en prévision de la prochaine propagation. La Figure 2.6 illustre les différentes types de nœuds pour un algorithme *one-lattice*. Bien qu'utilisant beaucoup moins de mémoire, Schulz *et al.* [62] et, plus récemment, Mattila *et al.* [61] ont mesuré des performances de calcul inférieures pour leur implantation du *one-lattice* comparativement à un *two-lattice*. Cette baisse de performance est attribuable à une moins bonne utilisation de la mémoire cache.

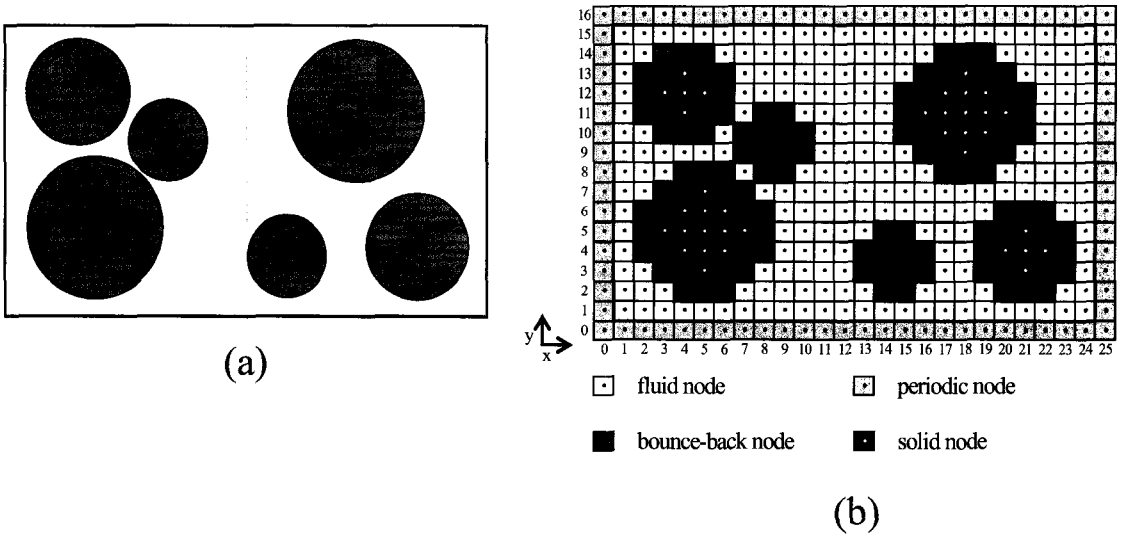


Figure 2.6 – Discrétisation schématique d'un milieu poreux (a) au moyen d'un domaine composé d'un réseau de 24×15 nœuds (b) pour un algorithme *one-lattice*.

La méthode de MBR-BGK utilisant l'algorithme *one-lattice* se résume donc aux quelques étapes suivantes :

1. Définition des propriétés du fluide (ν, ρ);
2. Définition des conditions frontières et discrétisation du domaine par une grille de type donné (δ_x et n_d);
3. Détermination de δ_t ou τ^* selon l'Équation (2.17);
4. Initialisation du temps : $t = 0$;
5. Initialisation du champ de vitesse $\mathbf{u}(\mathbf{x},0)$[23] et de la densité $\rho(\mathbf{x},0)=\rho$;
6. Calcul des populations à l'équilibre $f_i^{eq}(\mathbf{x},0)$ selon l'Équation (2.12);
7. Initialisation des populations $f_i(\mathbf{x},0)=f_i^{eq}(\mathbf{x},0)$;
8. Itération jusqu'à convergence du champ de vitesse;
    a. Calcul des variables hydrodynamiques macroscopiques ($\rho(\mathbf{x},t)$ et $\mathbf{u}(\mathbf{x},t)$) selon les Équations (2.13)[24];
    b. Calcul des populations à l'équilibre $f_i^{eq}(\mathbf{x},t)$ selon l'Équation (2.12)[24];

[23] En fonction des conditions initiales de l'écoulement : par exemple, $\mathbf{u}(x,t)=\vec{0}$ ou $\mathbf{u}(x,t)=\mathbf{U}$.

[24] Les étapes (8a) et (8b) ne sont pas nécessaires lors de la première itération puisque les étapes (5) et (6) en prennent déjà soin.

c. Calcul des populations post-collision et « prêtes à la propagation » en chaque nœud et dans les n_d directions du réseau:

$$f_i^*(\mathbf{x},t) = f_i(\mathbf{x},t) + \Omega_i(\mathbf{x},t), \quad i = 0,1,...,(n_d - 1); \tag{2.19a}$$

d. Imposition des conditions frontières périodiques et de rebond à mi-chemin (voir Annexe C);

e. Imposition de la perte de charge par une force externe (voir Annexe C);

f. Propagation des populations dans leurs directions respectives vers les nœuds voisins (en ordre de **x** descendant lorsque $\mathbf{e}_i$ pointe dans la direction **x** et vice-versa):

$$f_i(\mathbf{x} + \mathbf{e}_i\delta_t, t + \delta_t) = f_i^*(\mathbf{x},t), \quad i = 0,1,...,(n_d - 1); \tag{2.19b}$$

g. Incrémentation du temps : $t = t + \delta_t$;

9. Fin des itérations et impression des résultats (vitesse $\mathbf{u}(\mathbf{x},t)$ et pression relative $P(\mathbf{x},t)$) sachant que:

$$P(\mathbf{x},t) = c_s^2\left(\rho(\mathbf{x},t) - \langle\rho(\mathbf{x},t)\rangle\right) \quad \text{où } \langle\rho(\mathbf{x},t)\rangle = \rho\,. \tag{2.20}$$

Il est à noter qu'en pratique l'ordre entre collision et propagation n'a que peu d'importance puisque l'on itère autour des deux opérations. Certains chercheurs préfèrent propager d'abord et effectuer les collisions après. Toutefois, le reste des opérations doit s'effectuer dans la même séquence: par exemple, l'établissement des conditions frontières toujours après la collision, etc…

2.3.4 Algorithmes numériques récents

Récemment, de nouveaux algorithmes découlant du schéma C&P ont été proposés, à savoir les algorithmes *Lagrangian* [63], *shift* [64] et *swap* [65]. Ces algorithmes ainsi que les *one-lattice et two-lattice* ont été, très récemment, comparés par

Mattila *et al.* [61]. Ils ont montré que le shift et le swap sont supérieurs aux précédents en termes de performance de calcul (meilleur utilisation de la mémoire cache) et d'utilisation de la mémoire. L'algorithme *Lagrangian*[25] étudié a obtenu quant à lui les performances de calcul et d'utilisation de la mémoire les moins avantageuses. Pour cette raison, nous nous bornerons à décrire ici les algorithmes *shift* et *swap*.

L'algorithme *shift* ou à grille compressée découle du *two-lattice* dans le sens où il consiste à utiliser, conceptuellement parlant tout au moins, deux matrices comme ce dernier, mais à les faire se chevaucher en mémoire de telle façon que la dépendance spatiale des populations entre les deux pas de temps (c'est-à-dire les deux matrices) ne soit en aucunes circonstances rompue. L'idée est de ne conserver en tout temps en mémoire que le strict nécessaire, c'est-à-dire seulement les populations de l'itération précédente encore requises pour le calcul des nouvelles populations des nœuds restants à mettre à jour. En pratique, cela revient à utiliser une seule matrice additionnée d'une zone tampon et d'y déplacer alternativement à la fin ou au début les nouvelles populations. L'écriture séquentielle de ces dernières se fait dans un ordre inverse au déplacement en mémoire tel qu'illustré à la Figure 2.7. Étant donné la

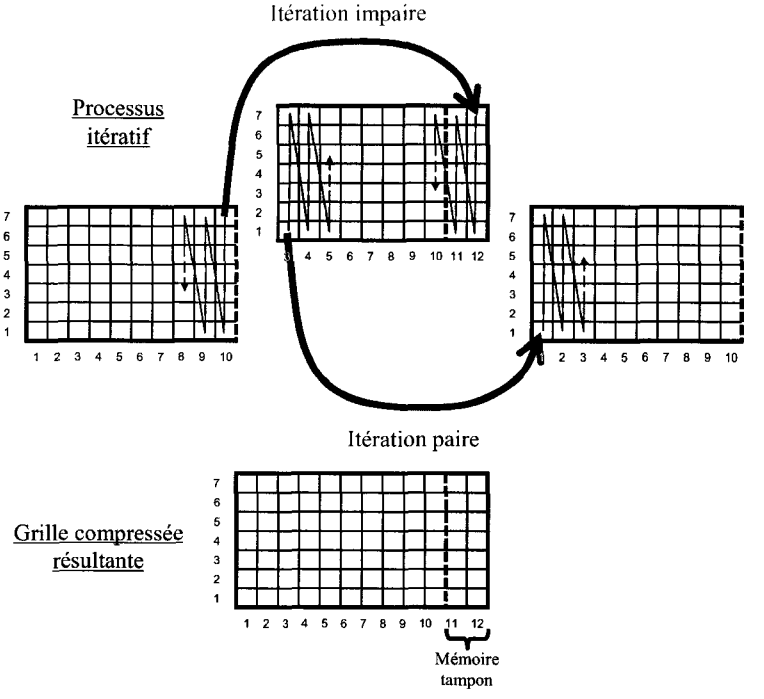


Figure 2.7 – Représentation 2D schématique de l'algorithme *shift*.

[25] Cet algorithme, comme son nom l'indique, consiste à considérer les populations dans un repère Lagrangien, ce qui permet d'éliminer la phase de propagation. Mais le gain en performance de calcul semble être éclipsé par des conversions de coordonnées entre repères lagrangien et eulérien plutôt gourmandes.

localité des règles de propagation (mise à jour à l'aide des plus proches voisins), la dépendance spatiale des populations (c'est-à-dire la distance en mémoire la plus grande entre deux nœuds voisins) ne dépasse pas deux colonnes sur le cas 2D présenté à la Figure 2.7[26].

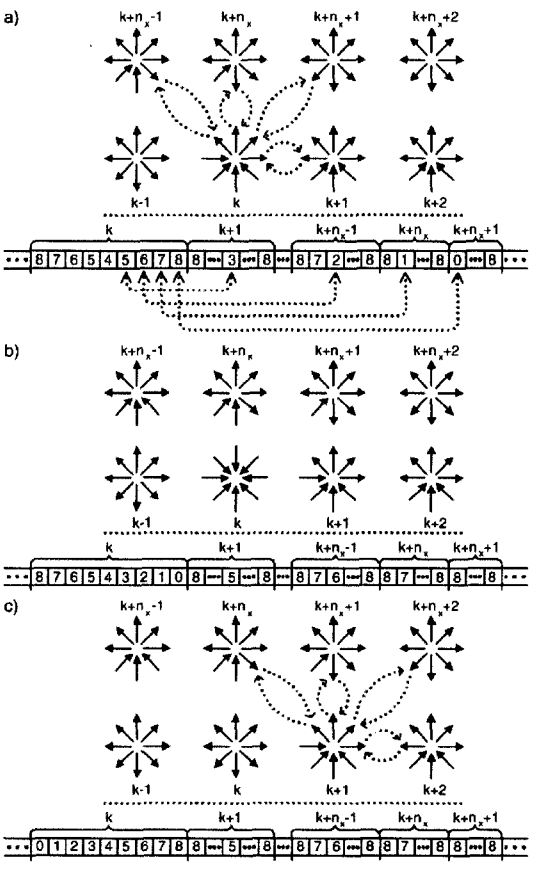


Figure 2.8 – Représentation schématique de l'algorithme *swap* pour un réseau D2Q9 [65].

Matilla *et al.* [65] ont complètement revisité le schéma C&P fusionné (Équation 2.18) en essayant de trouver une façon de propager les populations sans écraser les populations à l'itération précédente encore nécessaires et donc de s'affranchir du stockage de deux matrices (comme pour le *two-lattice*). L'algorithme *swap* a été ainsi nouvellement introduit. Tel qu'illustré à la Figure 2.8 pour un nœud donné k (situation (a)), il consiste à initialement permuter les populations uniquement avec les voisins qui n'ont pas encore été mise à jour avant d'exécuter la collision (situation (b)) à l'aide des populations nouvellement permutées et celles déjà obtenues. L'algorithme procède ensuite de la même manière avec le nœud k+1 (situation (c)). L'avantage de cet algorithme, au contraire du *one-lattice* et du *shift*, c'est qu'il ne requiert nullement l'addition d'espaces mémoire supplémentaires (pas de nœuds de rebond ou périodiques ou de zone tampon). Aussi simple que cet algorithme puisse paraître, il aura fallu plus de quinze ans pour mettre au point l'algorithme qui semble à première vue idéal!

[26] En fait, elle ne serait que d'une colonne et une cellule si ce n'était des conditions périodiques.

2.4 RÉDUCTION DE LA MÉMOIRE UTILISÉE PAR MBR-BGK

Nous allons aborder brièvement ici les méthodes disponibles dans la littérature permettant de réduire l'espace mémoire.

2.4.1 Schéma C&P simplifié

Martys & Hagedorn [66] ont proposé une simplification du schéma C&P fusionné en fixant τ^*=1. Le schéma de l'Équation (2.18) devient alors:

$$f_i(\mathbf{x},t) = f_i^{eq}(\mathbf{x}-\mathbf{e}_i\delta_t, t-\delta_t) \tag{2.21}$$

Outre une simplification des calculs, ce nouveau schéma à l'avantage de découpler complètement les nouvelles populations des anciennes. Seules les populations à l'équilibre local sont maintenant nécessaires et celles-ci peuvent être obtenues à partir de la vitesse et de la densité locales par l'entremise de l'Équation (2.12). La conséquence de tout ceci est qu'il n'est plus nécessaire de stocker en mémoire les populations, mais seulement les trois composantes de la vitesse et la densité. Pour un réseau D3Q15, le gain est substantiel car seulement 4 tableaux sont maintenant requis au lieu de 15. Le gain est encore plus substantiel avec des réseaux D3Q19 et D3Q27.

2.4.2 Structure de données de MBR-BGK

Dans les implantations traditionnelles de MBR, les données sur les populations sont stockées dans une ou deux matrices quadridimensionnelles selon le type d'algorithme utilisé (trois dimensions pour localiser la donnée dans l'espace et une dimension pour identifier la population dans l'espace des vitesses, soit F(i,j,k,b) où i, j et k sont les indices de position et b l'indice identifiant la population). Étant donné qu'aucun calcul n'est réalisé sur les nœuds solides, l'espace mémoire associé à ce type de nœud dans la/les matrice(s) est gaspillé. Autrement dit, les matrices obtenues sont creuses et le seront d'autant que la porosité du domaine sera faible. Pour remédier à ce gaspillage, plusieurs chercheurs [62,66,67-69] ont utilisé une technique de vectorisation des données souvent utilisée dans d'autres domaines dans les cas de matrices creuses. Ceci est illustré à la Figure 2.9. L'inconvénient est que la topologie fournie par la structure matricielle est maintenant perdue et qu'un adressage direct des nœuds n'est plus possible. Pour résoudre ce problème, on a recours à un adressage indirect au moyen d'une liste de connectivités pour chacun des nœuds. Bien sûr cette liste consomme de la mémoire et il existe une porosité seuil (environ 70-75% d'après [62,67-68]) au delà de laquelle le coût de cette liste devient prohibitif comparativement au gain en mémoire résultant de la vectorisation. En-dessous de cette valeur, les gains en mémoire sont croissants et deviennent rapidement conséquents.

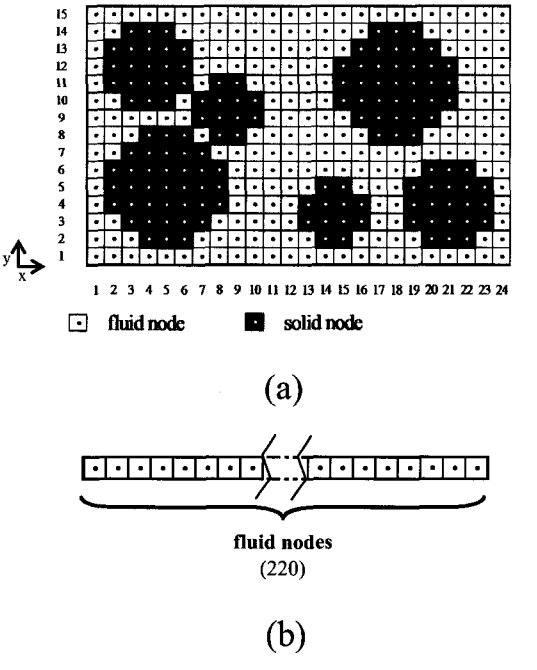


Figure 2.9 - Exemple 2D d'une vectorisation d'un domaine poreux.

2.5 AMÉLIORATION DE LA PERFORMANCE DE CALCUL DE MBR-BGK

Il existe de nombreuses façons d'améliorer l'exécution d'un code séquentiel MBR-BGK pour une architecture donnée :

1. l'optimisation du code par des techniques simples de programmation (ordre des boucles imbriquées, élimination des dépendances non nécessaires et des branchements, minimisation des opérations coûteuses comme les divisions, regroupement des données par bloc pour maximiser l'utilisation de la mémoire cache,...) [64-65,70-71];
2. le choix d'un algorithme plus approprié mettant, par exemple, plus à profit la mémoire cache ou minimisant le nombre d'opérations (voir Sections 2.3.3 & 2.3.4);
3. le choix d'une structure de données adaptée à l'algorithme utilisé;
4. l'utilisation d'artifices numériques pour accélérer la convergence vers une solution souhaitée.

Nous allons maintenant voir plus en détail les deux derniers points.

2.5.1 Impact de la structure de données

Un aspect important de la structure de données qui a été passé sous silence à la Section 2.4.2 est relié à la disposition des populations dans la structure de données même. En effet, stocker les populations comme F(i,j,k,b) ou F(b,i,j,k) ne revient pas au même. Si

la disposition n'a pas d'impact sur la quantité de mémoire utilisée, un choix judicieux peut permettre d'accélérer l'exécution du code par une meilleure utilisation de la mémoire cache lors de l'exécution des collisions ou des propagations. En Fortran, le format F(i,j,k,b) est dit « optimisé pour la propagation » alors que F(b,i,j,k) est dénommé « optimisé pour la collision ». Comme montré récemment par Wellein *et al.* [70] et Mattila *et al.* [61], de façon générale, la disposition optimisée pour la propagation semble donner de meilleurs résultats sur les différentes architectures scalaires testées (Xeon, Itanium 2 et Opteron), malgré que le gain obtenu dépende de l'architecture du système et de la taille du problème dans une certaine mesure. Finalement, Mattila *et al.* [61] ont proposé un version hybride des deux dispositions précédentes, appelé *bundle layout* ou disposition par paquet. Celle-ci semble donner de meilleures performances sur les architectures Xeon et Opteron testées.

2.5.2 Accélération de la convergence

Il est intéressant de remarquer que la méthode MBR-BGK telle qu'exposée jusqu'à présent permet en fait d'obtenir la solution transitoire des équations de Navier-Stokes entre les conditions initiales prescrites et la solution d'équilibre du système (solution stationnaire) et ne repose pas sur un processus itératif abstrait purement mathématique menant à la solution stationnaire. Toutefois, on est souvent seulement intéressé par la solution stationnaire du système comme c'est le cas si l'on désire calculer la perméabilité d'un milieu poreux. Dans ces conditions, il est conseillé de fournir des conditions initiales les plus proches possibles de la solution stationnaire pour accélérer la convergence. On en conviendra, cet estimé initial n'est pas toujours facile à prescrire, surtout pour les milieux poreux.

En conséquence, certains auteurs ont pensé à d'autres façons pour accélérer la convergence. Succi et ses coauteurs [72,73] ont proposé diverses approches, comme

l'élimination toute simple de la dérivée temporelle de l'équation de Boltzmann. Ce faisant, la différentiation obtenue se ramène à un schéma implicite amenant la résolution d'un système matriciel au moyen de solveurs itératifs avec préconditionneurs. Quoique ces méthodes réduisent significativement le temps de calcul sur un seul processeur, l'utilisation de solveurs itératifs complique d'autant la parallélisation de tels algorithmes et élimine ainsi un des grand attrait de MBR.

En revanche, dans le cas où une force externe est utilisée pour imposer un gradient de pression (voir Annexe C), Kandhai *et al.* [74] ont proposé d'utiliser une technique de relaxation itérative de la quantité de mouvement (RIQM) pour accélérer la convergence. L'idée est astucieuse : au lieu de garder la force externe constante telle que prescrite par l'Équation (C.4) (voir Annexe C), celle-ci est initialement accrue artificiellement et réajustée à intervalle de temps régulier vers la valeur prescrite[27]. Cette technique a pour effet d'accélérer la mise en mouvement du fluide et correspond en quelque sorte à augmenter initialement le gradient de pression et le réajuster progressivement vers la valeur voulue. La mise à jour de la force externe se fait comme suit :

$$\|\mathbf{f}\|_{k+1} = \|\mathbf{f}\|_k - (\Delta Q)_k \tag{2.22}$$

où k dénote le compteur d'itération de RIQM et ΔQ est la variation totale de quantité de mouvement dans la direction de **f** depuis la dernière itération en k. Le terme de droite de l'Équation (2.22) représente la perte moyenne de quantité de mouvement du fluide due aux pertes visqueuses. Grâce à cette technique, les auteurs rapportent des diminutions du nombre d'itérations nécessaires pour atteindre l'équilibre de 45 à 97% selon les cas! De plus, le gros avantage de cette stratégie, c'est qu'elle ne remet nullement en cause le

[27] Dans [74], la force externe est augmentée initialement par un ou plusieurs ordres de grandeur et rajustée tous les 50 pas de temps.

caractère explicite de MBR et donc conserve pratiquement intégralement son efficacité à la parallélisation[28]. On aura donc tout intérêt à mettre à profit cette technique.

2.6 MÉTHODES DE MBR-BGK ADAPTATIVES

Comme nous avons pu le voir, la méthode standard MBR-BGK permet seulement aux populations de particules de passer d'un nœud de la grille à un nœud adjacent distant de δ_i dans un intervalle de temps δ_t ($\delta_i = \|\mathbf{e}_i\| \delta_t$). Autrement dit, les discrétisations en temps et en espace sont assujetties à la discrétisation en vitesse de l'équation de Boltzmann. Cette contrainte implique donc que la grille sous-jacente soit uniforme et structurée puisque les vitesses $\mathbf{e}_i$ sont constantes. Toutefois, pour une meilleure gestion des ressources informatiques, des grilles non-uniformes et/ou non-structurées sont très souvent utilisées en mécanique des fluides numérique classique. De nombreux chercheurs [75-92] se sont donc attachés à relaxer cette contrainte pour MBR-BGK. On distingue principalement quatre grandes classes de méthodes qui ont été développées dans cette optique :

1. Méthodes multi-échelles ou à grilles embarquées;
2. Méthodes MBR aux différences finies;
3. Méthodes MBR aux volumes finis;
4. Méthodes d'interpolation.

[28] Dans le cadre d'une parallélisation par décomposition de domaine, le calcul de la nouvelle force externe devrait être centralisé sur un processeur qui assemblera toutes les variations locales de quantité de mouvement au moyen d'opérations MPI « gather » ou « reduce » avant de distribuer la nouvelle valeur à tous les autres processeurs (« broadcast »). Cette opération de mise à jour, quoique non parallèle à la base donc peu enclin à l'équilibrage de tâches, n'est toutefois réalisée que sporadiquement comparée aux autres opérations et ne requiert que peu de communications. Elle ne devrait donc réduire l'efficacité de la parallélisation que de façon marginale tout en procurant un gain substantiel en temps de calcul.

Ces différentes méthodes constituent en fait de nouvelles implantations de MBR-BGK à part entière. Nous n'allons pas les examiner ici mais plutôt faire ressortir pour chacune d'entre elles leurs avantages et/ou leurs éventuels inconvénients. Nous référons plutôt le lecteur intéressé par plus de détails à l'Annexe E pour une revue plus consistante.

À cause des niveaux hiérarchiques inhérents aux méthodes multi-échelles [75-79], celles-ci se portent mal à la parallélisation par décomposition de domaine ou tout du moins à l'équilibrage de tâches sur les processeurs. Les méthodes MBR aux différences finies [80-81] sont quant à elles peu pratiques car elles nécessitent la détermination de transformations géométriques à appliquer à une grille cartésienne pour se conformer au domaine poreux. En fait, seules les méthodes MBR aux volumes finis [82-86] et les méthodes d'interpolation [87-92] sembleraient être de bonnes candidates car elles conservent le parallélisme de MBR tout en étant, à première vue, conceptuellement simples à appliquer aux milieux poreux. Pourtant, une analyse plus poussée nous amène à croire que les gains en ressources informatiques escomptés ne seraient peut-être pas au rendez-vous pour les raisons suivantes: 1) de nombreux calculs additionnels sont requis en chacun des nœuds; 2) une quantité non-négligeable de mémoire supplémentaire est nécessaire (au moins 56 octets/nœud); et/ou 3) une méthode de triangulations de Delaunay pour générer les volumes finis doit être utilisée[29,30]. Bien sûr, l'accroissement du nombre d'opérations et de la quantité de mémoire nécessaire peut éventuellement être compensé par la réduction du nombre de nœuds. Ce qui nous amène à considérer une difficulté encore plus importante et reliée à la physique du problème : sur quels critères générer un maillage adaptatif ou un nuage de points d'interpolation dans un milieu poreux ?

La première réponse qui peut nous venir à l'esprit consisterait à raffiner le maillage dans les petits pores du domaine et à les grossir ailleurs. Toutefois, la mécanique des fluides nous apprend qu'un fluide s'écoule dans un milieu poreux de façon préférentielle

[29] Or générer un maillage adaptatif sur une géométrie aussi complexe pour des centaines de millions, voire des milliards de nœuds, demeure encore de nos jours un problème.

dans un petit nombre de capillaires de moindres résistances (c'est ce que l'on appelle le « renardage » ou « *channeling* » en anglais) et non à travers les plus petits pores! Il faudrait donc pouvoir raffiner seulement dans ces capillaires, lesquels ne sont pas connus a priori. La technique précédemment mentionnée en viendrait donc à augmenter inutilement le nombre de nœuds là où ils ne seraient pas nécessaires et/ou à en enlever là où ils le seraient, soit un temps de calcul plus long pour une moins bonne précision! Notez que l'argumentaire ici développé est strictement valide pour des écoulements monophasiques dans des milieux saturés car, pour des écoulements polyphasiques, les phénomènes de tension de surface font que les écoulements se produisent de façon préférentielle dans les plus petits capillaires. On voit donc bien que le maillage adaptatif se doit de prendre en compte la physique du problème.

2.7 RÉSUMÉ

Comme nous avons pu nous en rendre compte au cours de ce chapitre, MBR standard semble être une méthode très efficace pour la résolution des équations de Navier-Stokes pour des géométries complexes dans les limites de faibles nombres de Mach et Knudsen. Cependant, sur un seul processeur pour des géométries simples, elle est en pratique au moins aussi gourmande en ressources informatiques que les méthodes classiques de mécanique numérique des fluides [11,28,47,93]! En fait, les grands avantages de la méthode de MBR résident dans sa simplicité, sa facilité à discrétiser adéquatement le domaine et son efficacité à la parallélisation grâce au schéma explicite de la méthode et à sa grande localité[31]. MBR est donc une méthode faite pour être implantée en parallèle et, dans ces conditions, elle surpasse de loin en efficacité toutes les

[30] Pour générer le nuage de points pour les méthodes d'interpolation, une méthode d'octree peut être mise à profit en lieu et place d'une triangulation de Delaunay.

[31] La localité de la méthode est un grand avantage, car le principe de réutilisabilité des données promu par les architectures modernes basées sur la mémoire cache peut être facilement mis à profit.

méthodes classiques. Toutefois, son développement heuristique a engendré une grande variété d'algorithmes et de méthodes d'efficacités diverses qui procurent souvent chez le néophyte une certaine confusion. C'est ce que nous avons essayé d'éclaircir au cours de ce chapitre. Dans une optique de meilleure gestion de la mémoire, les algorithmes C&P *one-lattice*, *shift* et *swap* sont supérieurs au *two-lattice* originel. De plus, le choix judicieux d'une structure de données vectorielle (optimisée pour la propagation) dans le cadre de simulations dans des milieux à grandes porosités ($\varepsilon < 70\text{-}75\%$) peut procurer une réduction additionnelle significative de la mémoire et une meilleure exécution du code grâce à une utilisation accrue de la mémoire cache. Finalement, bien qu'à première vue attractives, les méthodes MBR-BGK à grilles adaptatives se trouvent limitées par un manque de parallélisme et/ou des difficultés à générer des discrétisations appropriées du milieux poreux.

À y regarder de plus prêt, on peut être pantois du tour de force réalisé : nous avons remplacé les équations non-linéaires de Navier-Stokes aux dérivées secondes décrivant l'écoulement d'un fluide à l'échelle macroscopique par une discrétisation explicite aux différences finis de l'équation de Boltzmann aux dérivées simples décrivant le transport de particules à l'échelle mésoscopique : il fallait y penser ! La phase de propagation étant une opération purement linéaire, la non-linéarité de MBR est en fait dissimulée dans l'opérateur de collision. En théorie, MBR peut donc ne pas se limiter à de faibles nombres de Reynolds. Cette limitation est purement due au manque de stabilité du schéma explicite de discrétisation. Donc, un schéma implicite et/ou d'ordre supérieur peut permettre d'atteindre des nombres de Reynolds élevés mais au détriment toutefois du parallélisme.

En ce qui concerne le parallélisme, nous nous efforcerons dans le prochain chapitre de préciser les aspects parallèles de MBR : pourquoi MBR est si efficace en parallèle et quelles ont été les stratégies employées et celles qui sont envisagées pour remédier au mauvais équilibrage de tâches résultant de l'hétérogénéité du domaine.

CHAPITRE 3

ASPECTS PARALLÈLES DE LA MÉTHODE DE BOLTZMANN SUR RÉSEAU

En quoi consiste le parallélisme au juste ? L'idée du parallélisme n'est en fait pas récente. En 1637, René Descartes publie son célèbre traité sur « Le discours de la méthode ». Parmi les quatre préceptes que Descartes expose « *pour bien conduire sa raison et chercher la vérité dans les sciences* », le deuxième définit en quelque sorte les bases du parallélisme : « [je conviens] *de diviser chacune des difficultés que j'examinerais en autant de parcelles qu'il se pourrait, et qu'il serait requis pour les mieux résoudre* ». Pour remonter encore plus loin dans l'histoire, les Romains de la *Pax Romana* avaient bien compris que pour mieux administrer leur vaste empire, mieux valait le scinder en de nombreuses provinces plus faciles à gérer individuellement, soit le célèbre « *divide et impera* » (diviser pour régner). Le parallélisme consiste donc à subdiviser un énorme problème en problèmes de tailles plus réduites, les résoudre de façon concourante et combiner les solutions partielles pour obtenir la solution globale du problème initial. En théorie, si un problème est scindé en n_p parties équivalentes et chacune d'elles résolue sur un processeur de façon concourante, le temps pour atteindre la solution du problème devrait donc être divisé par un facteur n_p.

Toutefois, en pratique, peu de problèmes se prêtent à ce scénario idyllique. Pour qu'une application y parvienne, elle doit en fait remplir un certain nombre de critères quant au :

- contenu parallèle ou « parallélisable »,
- rapport entre calcul et communication,

- et équilibrage de tâches.

Une revue exhaustive de la littérature sur MBR montre que, malgré la nature parallèle de MBR, en fait un nombre limité d'articles traitent réellement des aspects parallèles de la méthode [62,67,68,75,94-101]. Au cours de ce chapitre, nous examinerons donc comment MBR se comporte du point de vue des trois critères précédemment cités et quelles stratégies ont été ou peuvent être employées pour améliorer les performances de la méthode.

3.1 CONTENU PARALLÈLE

Un programme est en général composé d'une partie parallèle et d'une partie non parallélisable ou séquentielle. On définit le facteur d'accélération (A) et l'efficacité (E) d'un programme parallèle comme suit :

$$A(n_p) = \frac{t_s}{t_p} = \frac{t_s}{f_s t_s + (1-f_s) t_s / n_p} = \frac{n_p}{1+(n_p-1) f_s} \tag{3.1}$$

et

$$E(n_p) = \frac{t_s}{t_p \times n_p} = \frac{A(n_p)}{n_p} \tag{3.2}$$

où f_s représente la fraction des calculs qui ne peut être parallélisée, n_p le nombre de processeurs, t_s le temps d'exécution du programme séquentiel, t_p le temps d'exécution du programme parallèle. L'Équation (3.1) est connue sous le nom de loi d'Amdahl [51] et mesure le rapport entre le temps de calcul en séquentiel et celui en parallèle. Lorsque le nombre de processeurs tend vers l'infini, on a $\lim_{n_p \to \infty} A(n_p) = \frac{1}{f_s}$. On voit donc que même avec

un nombre infini de processeurs, l'accélération maximale que l'on peut obtenir est limitée à $1/f_s$. La Figure 3.1 illustre ce point. Un algorithme dont la fraction séquentielle est importante ne pourra donc pas tourner efficacement sur un ordinateur parallèle. Notons cependant que la loi d'Amdahl suppose que la taille du problème demeure constante, donc que la fraction séquentielle f_s demeure constante. En règle générale, plus le problème est gros, plus la fraction séquentielle est faible et donc plus efficace devrait être l'algorithme pour un nombre donné de processeurs[32].

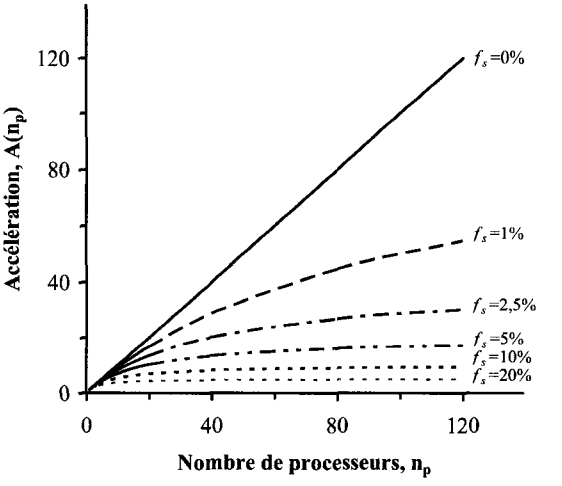


Figure 3.1 – Variation de l'accélération en fonction de la fraction séquentielle d'après la loi d'Amdahl.

Si l'on examine de près l'algorithme C&P (page 26) utilisé par MBR, on peut constater que la fraction séquentielle de MBR peut se résumer à la lecture initiale des données d'entrée (δ_x, δ_t, ν, ρ, etc) et possiblement à l'écriture finale des résultats par un processeur. En fait, d'après [47], c'est 99% du temps de calcul qui est occupé par la phase de collision (étapes 8a-c de l'algorithme) qui est une opération purement locale donc parfaitement parallélisable. Ainsi, en pratique, la fraction séquentielle est nettement inférieure à 1%. Donc, l'algorithme devrait conserver une bonne accélération pour un nombre de processeurs bien au-delà de la centaine. Succi rapporte d'ailleurs une très bonne évolutivité jusqu'à au moins 512 processeurs pour des simulations MBR d'écoulement turbulent dans une conduite [47,99]. Toutefois, la loi que nous venons de décrire prend juste en compte le contenu séquentiel du code, mais ne traitent pas des communications qui peuvent être à leur tour un facteur limitant...

[32] Bien sûr, nous n'avons pas pris en compte pour le moment le coût des communications.

3.2 COMMUNICATIONS

Dans tout programme parallèle, le maître mot est de minimiser le coût des communications ou de maximiser le rapport temps de calcul sur temps de communication, c'est-à-dire maximiser la granularité[33] tout en maintenant un parallélisme satisfaisant. En effet, si la granularité devient trop faible, alors les processeurs passent plus de temps à s'échanger de l'information qu'à calculer. Ceci se caractérise par une baisse de l'accélération lorsque l'on accroît le nombre de processeurs alloués à un problème donné (cf. Figure 3.2). On a ce que l'on appelle un surdébit de communication. Ce problème est aussi aggravé par des considérations purement technologiques : en effet, les communications inter-processeurs sont de plusieurs ordres de grandeur plus lentes que les opérations arithmétiques au niveau du processeur. Il faut donc pouvoir faire un bon nombre de calculs arithmétiques pour pouvoir compenser les communications. Le temps de communication pour un transfert de données (t_{com}) se subdivise en général comme suit :

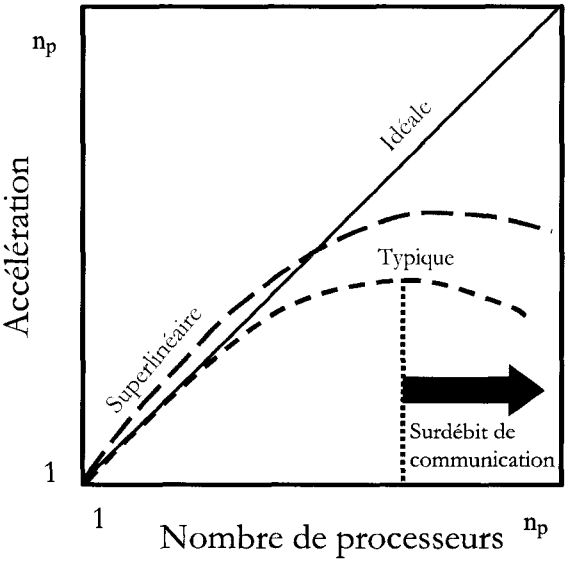


Figure 3.2 – Effet d'un surdébit de communication sur l'accélération.

$$t_{com} = t_{lat} + n \times t_{don} \tag{3.2}$$

où t_{lat} est le temps de démarrage ou de latence (c'est le temps nécessaire pour initier une communication), t_{don} est le temps requis pour transmettre une donnée (un mot) et n le nombre de mots à envoyer. Pour diminuer l'impact éventuel de la latence, on préférera

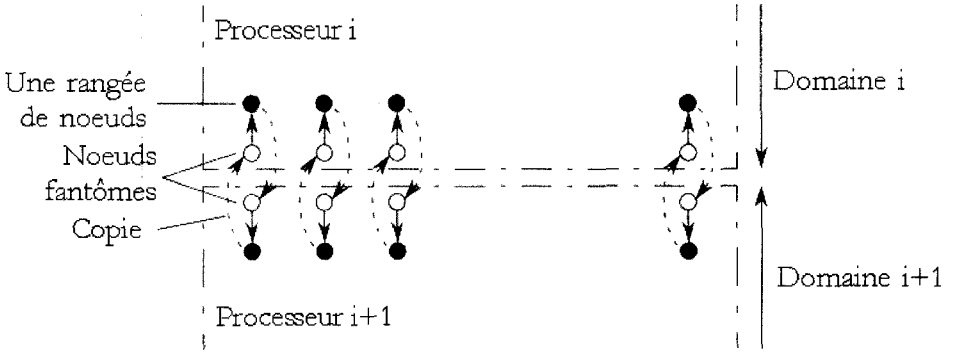


Figure 3.3 – Transfert des données d'un processeur à un autre au moyen de zones fantômes [51].

regrouper les données à envoyer dans un seul vecteur au lieu de les envoyer individuellement une à la fois. De plus, l'architecture du système de communication de l'ordinateur parallèle a son importance : lorsque possible, on privilégiera par exemple les communications directes pour éviter le cumul de latence lors du transit de l'information d'un commutateur à un autre.

Du point de vue du rapport entre communications et calculs (que nous appellerons dorénavant r_{cc}), MBR offre en théorie de très bonnes performances permettant ainsi une parallélisation efficace. Les seules communications requises proviennent en fait de la phase de propagation, laquelle transfère l'information d'un nœud donné vers les nœuds voisins immédiats. Dans le cadre d'une décomposition de domaine, seule l'information des nœuds aux frontières devra donc être échangée de part et d'autres de la frontière et copiée dans des zones fantômes[34] (Figure 3.3). D'après Succi [47], les méthodes classiques de différences finies sont beaucoup moins efficaces que MBR en terme de parallélisation à cause d'une moins bonne granularité. En conséquence, le caractère local et coûteux en temps de calcul de la phase de collision combinée à un débit de communication relativement faible procure à MBR des caractéristiques presque idéales

[33] On définit la granularité comme étant la quantité de calculs effectués entre deux points de communication.

[34] En fait, nul n'est besoin de copier toutes les populations des noeuds frontières, seules les populations pointant dans la direction du transfert sont nécessaires.

pour la parallélisation. Toutefois, un dernier critère doit aussi être rempli pour assurer une parallélisation idéale : il s'agit, comme nous l'avons déjà mentionné, de l'équilibrage des tâches entre les processeurs. Examinons donc plus en profondeur ce concept.

3.3 ÉQUILIBRAGE DE TÂCHES

Un mauvais équilibrage de tâches résulte d'une mauvaise répartition de la charge de calculs entre les processeurs. Le temps global d'exécution du programme correspond alors à la tâche la plus coûteuse. Il convient donc d'équilibrer la charge entre les processeurs (cf. Figure 3.4). Les problèmes d'équilibrage de tâches pour MBR sont dus soit à la complexité du domaine de simulation considéré, soit à l'hétérogénéité éventuelle de l'ordinateur parallèle[35]. Ils n'ont rien à voir en fait avec les propriétés intrinsèques de la méthode. Toutefois, les différentes stratégies possibles d'équilibrage peuvent s'adapter de façons diverses en fonction de ses propriétés. Ainsi, une simple décomposition de domaine par tranches sur un ordinateur parallèle homogène pourra s'avérer plus que satisfaisante pour le calcul de l'écoulement dans une conduite relativement simple. Mais cette même stratégie s'avérera inefficace dans le cas de milieux poreux ou de domaines dont la géométrie est évolutive, tel que souligné par Pan *et al.* [68]. Toutefois, vu la simplicité de sa structure de données, MBR est

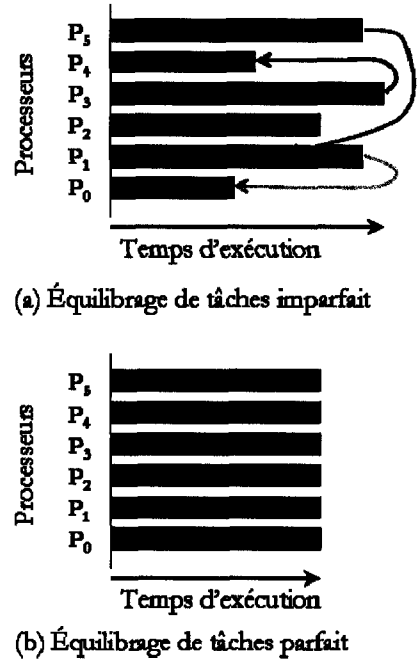


Figure 3.4 – Rééquilibrage de tâches.

[35] C'est-à-dire un ordinateur parallèle dont les processeurs ne sont pas uniformes en terme de performance.

relativement bien armée pour résoudre efficacement les problèmes de mauvaises répartitions de charge.

Examinons maintenant les différentes stratégies de parallélisation qui pourraient permettre de préserver une bonne performance dans le cadre de la résolution des écoulements dans les milieux poreux par MBR. Nous les avons regroupés dans trois principales familles reposant sur diverses techniques de décomposition de domaine: 1) les décompositions classiques cartésiennes, 2) les décompositions avancées minimisant ou non les communications, et 3) une technique de décomposition récemment proposée que nous appellerons « partition vectorielle équitable».

3.3.1 Décompositions de domaines cartésiennes

Par cette stratégie, le domaine de calcul est divisé en sous-domaines de façon géométrique, soit en tranches, en blocs ou en cubes (Figure 3.5). Cependant, toutes les divisions géométriques du domaine ne sont pas équivalentes d'un point de vue de l'efficacité. On aura en général avantage à :

- minimiser le rapport surface/volume des sous-domaines de façon à maximiser la granularité. On préférera donc ici des

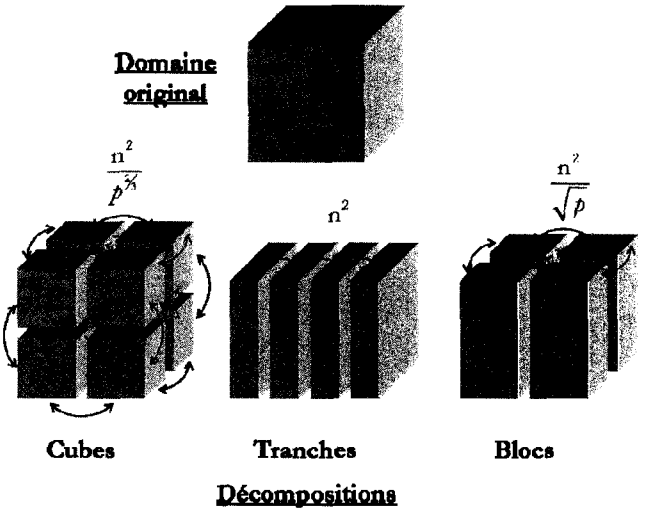


Figure 3.5 – Différentes décompositions de domaine possibles par inspection pour un domaine de taille n^3 et communications associées en rouge (les expressions figurant en rouge représentent la charge de communications pour les flèches pleines rouges et *p* représente ici le nombre de processeurs).

décompositions par cubes ou par blocs plutôt que par tranches;

- minimiser le nombre de domaines voisins à qui l'on doit communiquer de l'information pour diminuer le nombre d'opérations send/receive (flèches rouges de la Figure 3.5). Toutefois, la quantité d'informations à communiquer est en général plus grande pour un nombre moins grand de voisins. Il existe donc une configuration optimale qui dépend des temps de latence et de transfert de données (cf. Figure 3.6). De façon générale, on préférera les décompositions par tranches pour des temps de latence long et par blocs pour des temps courts.

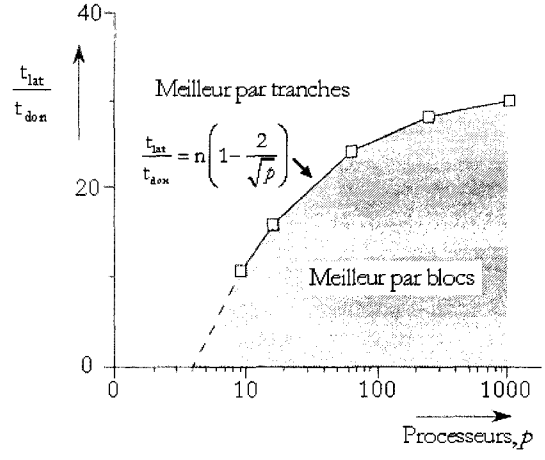


Figure 3.6 – Diagramme pour le choix d'une décomposition optimale bidimensionnelle (d'après [51]).

Pour tout domaine, il y a donc une décomposition optimale qui dépend avant tout des caractéristiques de l'ordinateur parallèle (nombre de processeurs et leur vitesse, temps de latence, largeur de bande) ainsi que du traitement vectoriel de l'ordinateur et/ou du compilateur (Fortran fonctionnant à l'inverse de C dans ce domaine). Ainsi, on trouve dans la littérature des résultats qui semblent à première vue contradictoires : pour des simulations MBR, Kandhai *et al.* [49,74] ont montré numériquement et théoriquement la supériorité des décompositions par blocs. La même année, Satofura & Nishioka [97] ont trouvé qu'une décomposition par tranches horizontales était plus efficace que des décompositions par blocs et par tranches verticales.

Néanmoins, ce genre de décomposition pour des domaines hétérogènes (tels que les milieux poreux) possèdent une faible évolutivité car, à mesure que le nombre de processeurs augmente, la variabilité de la porosité entre les sous-domaines devient plus grande et il en résulte une mauvaise répartition de la charge.

3.3.2 Décompositions de domaine avancées

Ce genre de stratégies utilise des algorithmes d'optimisation avancés de façon à répartir *a priori* les nœuds du maillage et se réfère au problème de partition de graphes. Malheureusement, ce type de problème est reconnu comme étant de type N-P complet[36]. Par conséquent, pour trouver des solutions acceptables, les chercheurs ont recours à des méthodes heuristiques qui ne peuvent fournir que des solutions sub-optimales. La littérature scientifique sur le sujet est très exhaustive et nous nous bornerons ici à en donner les grandes lignes. Parmi les méthodes heuristiques disponibles, mentionnons les méthodes d'heuristiques gourmandes (« greedy heuristics »), de recuits simulés, de réseaux neuronaux, d'algorithmes génétiques, d'algorithmes multi-niveaux, de bissections récursives orthogonales et spectrales. Des critères purement géométriques ou des fonctions de coût basées sur la densité des nœuds et la densité des connectivités (communications) entre les sous-domaines sont souvent employées. Parmi les plus couramment utilisés en relation avec l'équilibrage de tâches, mentionnons :

1. Le Recuit Simulé (RS): une fonction de coût est directement minimisée par un processus s'apparentant au refroidissement physique lent ;
2. La Bissection Récursive Orthogonale (BRO): une méthode simple qui divise le domaine en deux par un plan vertical, puis coupe de nouveau en deux chacune des moitiés cette fois-ci au moyen d'un plan horizontal, puis chacun des quarts de nouveau

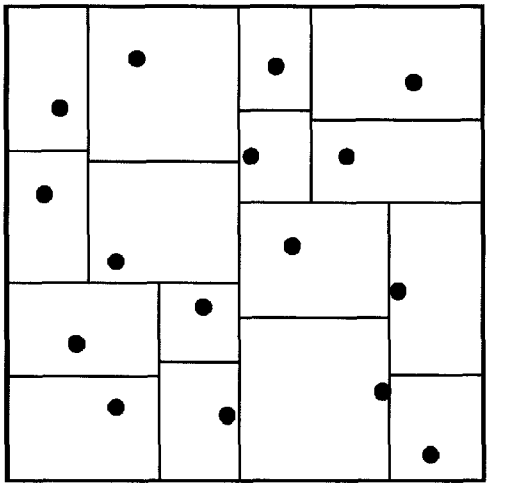


Figure 3.7 – Exemple de bissection récursive orthogonale.

[36] N-P signifie non polynomial, c'est-à-dire qu'il n'existe probablement aucun algorithme polynomial en temps capable de le résoudre !

verticalement et ainsi de suite (Figure 3.7). La position des plans est motivée en général par des considérations géométriques (centre de masse des nœuds). Cette méthode crée des patrons de communication qui sont en fait en général assez compliqués et donc ne minimise pas forcément les communications entre processeurs. Toutefois, dans une géométrie non-uniforme mais relativement simple, Kandhai *et al.* [74] ont obtenu une très bonne répartition de la charge en comparaison avec celles obtenues par des décompositions par tranches ou blocs. ;

3. La Bissection Récursive Spectrale (BRS): cette méthode utilise la même technique de base de partition que BRO, mais la localisation du plan de coupe est réalisée au moyen d'un vecteur propre d'une matrice de structure équivalente à la matrice de connectivité du domaine. Cette méthode tient donc compte des communications.

Dans le cas où minimiser le temps de calcul est plus important que minimiser le temps de communication, BRO est efficace car relativement rapide. Les autres méthodes sont beaucoup plus intensives en terme de temps de calcul et leur utilisation devrait se limiter au cas où les communications sont le facteur limitant. Toutefois, leur évolutivité laisse grandement à désirer car le problème à la base étant N-P complet, le temps de calcul s'accroît très rapidement avec le nombre de sous-domaines à trouver.

De nombreux logiciels commerciaux et librairies utilisant pour la plupart des algorithmes multi-niveaux sont disponibles pour résoudre les problèmes de partition de graphes. Ceux-ci, en plus d'équilibrer la tâche, réduisent les communications à l'aide d'une fonction de coût minimisant les connectivités entre les sous-

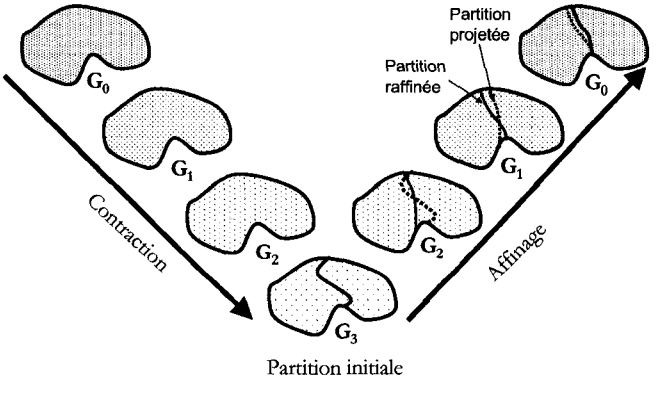


Figure 3.8 – Principe de l'algorithme multi-niveaux.

domaines. Citons en autres Chaco, Jostle, METIS (et sa version parallèle ParMETIS), Scotch, Plum et Party [102-104]. Les algorithmes multi-niveaux réduisent le temps de partition considérablement par une procédure constituée de trois phases : 1) une phase de contraction par laquelle on rend successivement le graphe original plus grossier (G_0 à G_3 à la Figure 3.8) de manière à réduire le nombre de nœuds tout en conservant des caractéristiques similaires ; 2) une phase de partition, par exemple, au moyen d'une bissection spectrale du graphe le plus fin (G_3) ; 3) une phase d'affinage par laquelle la partition initiale est successivement rétro-propagée par une série de projections et de raffinements jusqu'au graphe initial (G_o). Il existe de nombreuses façons de réaliser la contraction[37]. Pour l'affinage des projections, l'algorithme heuristique de Kernighan-Lin est souvent employé [105].

Schulz *et al.* [62], Dupuis & Chopard [67] et, plus récemment, Axner *et al.* [94] ont appliqué METIS avec succès dans le cadre de simulations MBR mais pour décomposer des domaines de tailles relativement modestes. Axner *et al.* [94] rapporte des temps de calcul de l'ordre 1,3-2,5 min pour obtenir une partition d'un million de nœuds. La complexité de METIS étant $O(n + m + k \log(k))$ [106], où n est le nombre de nœuds, m le nombre d'arêtes et k le nombre de partitions désirées, il est peu probable de pouvoir réaliser en un temps acceptable des décompositions de plusieurs centaines de millions de nœuds, encore moins de milliards, et ceci même en utilisant une version parallèle comme ParMETIS. Autres facteurs à la décharge des techniques de partition de graphes: leur coût mémoire prohibitif, des patrons de communications souvent compliqués et les fonctions de coût de communication souvent dépendantes de l'architecture utilisée et difficile à évaluer.

[37] Des algorithmes de regroupement de nœuds comme « k-means » pourraient par exemple être employés.

3.3.3 Partition vectorielle équitable

Récemment, Wang *et al.* [95] ont proposé une technique relativement simple et à faibles coûts en mémoire et en calculs pour équilibrer les tâches. Elle consiste à mettre à profit une structure de données vectorielle telle que proposée précédemment pour réduire les coûts en mémoire de MBR (voir Section 2.4.2, page 31). Le vecteur de données global ainsi obtenu est tout simplement divisé en sous-vecteurs de tailles égales, chacun desquels étant ensuite assigné à un processeur spécifique, tel qu'illustré à la Figure 3.9. Cette technique permet d'obtenir un équilibrage de tâches optimal et offre un patron de communication simple, puisque chaque processeur n'a besoin de communiquer qu'avec deux voisins. Toutefois, les interfaces entre processeurs ne sont pas nécessairement droites (interface en escalier) et le schéma optimal des populations à envoyer aux processeurs voisins est nettement plus compliqué. Par souci de simplicité, Wang *et al.* [95] ont choisi d'envoyer l'ensemble des populations des nœuds à l'interface vers les processeurs voisins, mais cette façon de faire est préjudiciable aux

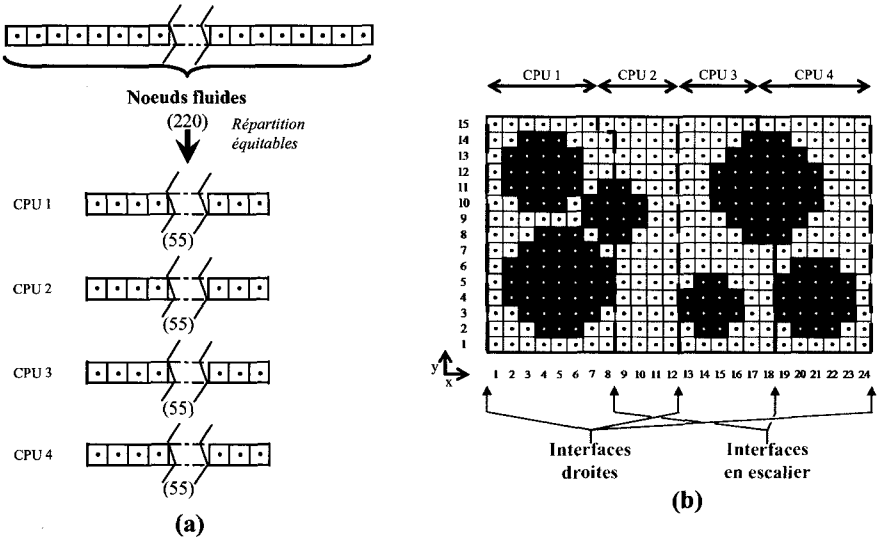


Figure 3.9 – Représentation 2D schématique de la partition vectorielle équitable sur 4 processeurs. Le vecteur de donnée est divisé en sous-vecteurs de tailles égales, chacun desquels étant assigné à un processeur spécifique (a). En (b), la répartition des sous-domaines obtenues.

communications et donc à la performance parallèle. De plus, l'algorithme MBR utilisé, soit le *two-lattice*, ne procure pas a priori un gain en mémoire optimal.

3.4 RÉSUMÉ ET OBJECTIFS SPÉCIFIQUES

Au cours des deux derniers chapitres, nous avons pu voir que la méthode de Boltzmann sur réseau est une méthode très efficace pour résoudre les écoulements de fluides dans les milieux poreux et qu'elle se prête merveilleusement à la parallélisation grâce à son fort contenu parallèle et sa granularité plus importante que les méthodes traditionnelles de mécanique des fluides numérique. Toutefois, la nature hétérogène des milieux poreux implique un déséquilibre de la charge lorsque les simulations sont réalisées en parallèle sur un grand nombre de processeurs au moyen de décompositions de domaine classiques. De nombreuses stratégies ont été proposées en ce qui a trait à l'algorithmique de la méthode de façon à mieux gérer les ressources informatiques tant sur le plan séquentiel que parallèle. Des techniques de décompositions de domaine avancées telles la bissection récursive orthogonale ou des techniques de partition de graphes multi-niveaux ont été utilisées, mais procurent des patrons de communication très complexes et/ou sont limitées à des systèmes de tailles relativement modestes (~1-10 million de nœuds). La méthode de partition vectorielle équitable récemment proposée semble sur ces points beaucoup plus flexible, en plus d'être relativement simple à implanter. Cependant, par souci de réduction des communications, un schéma optimal des populations à transférer d'un processeur à l'autre serait nécessaire. Pour finir, une application parallèle se doit dans la mesure du possible d'être la plus évolutive et la plus portable possible pour être performante. On doit pouvoir conserver une très bonne efficacité lorsque la taille du problème (évolutivité reliée à l'algorithme) et/ou de

l'ordinateur parallèle[38] (évolutivité reliée à l'architecture) est augmentée. On voudrait aussi pouvoir conserver une bonne efficacité quel que soit le type d'ordinateur, c'est ce que l'on appelle la portabilité. L'établissement d'un modèle théorique de performance parallèle fidèle aux mesures permet de juger de l'évolutivité et de la portabilité d'un code.

Par conséquent, dans un souci de meilleure gestion des ressources informatiques, des combinaisons judicieuses des meilleurs algorithmes MBR et de techniques de décompositions de domaine sont proposées, améliorées et testées dans le cadre de la résolution d'écoulement monophasique dans des milieux poreux complexes. Spécifiquement, nous comptons:

1. Concevoir un algorithme *one-lattice* combiné à une méthode de partition vectorielle équitable utilisant un schéma optimal des populations à transférer entre processeurs et tester ces performances parallèles et son utilisation de la mémoire. À ces fins, des modèles théoriques de performance parallèle et d'utilisation de la mémoire seront établis et les performances obtenues seront aussi comparées à un algorithme *one-lattice* combiné à une décomposition de domaine cartésienne classique. Les tests seront réalisés sur deux grappes de calcul distinctes à mémoire distribuée;

2. Concevoir un algorithme *shift* combiné à un schéma C&P simplifié (avec $\tau^*=1$) et une méthode de partition vectorielle équilibrée et tester les performances parallèles et son utilisation de la mémoire. Les performances ainsi obtenues seront comparées avec les modèles et algorithmes établis à l'étape 1 sur les mêmes grappes de calcul. La limite de validité de cet algorithme sera clairement définie;

3. Appliquer l'algorithme développé à l'étape 1 sur un cas d'étude grandeur nature, soit la simulation d'écoulement d'un fluide newtonien monophasique dans des entassements compressés de sphères de différentes polydispersités, et comparer les perméabilités ainsi obtenues numériquement avec des données expérimentales

[38] C'est-à-dire, le nombre de processeurs.

et la corrélation de Carman-Kozeny (Équation (1.2), page 3). Finalement, proposer une corrélation de Carman-Kozeny modifiée qui permet de mieux prédire les écoulements dans les milieux poreux et ainsi tenir compte de la polydispersité des éléments qui les constituent.

Les travaux reliés aux deux premiers objectifs spécifiques sont présentés respectivement sous forme d'articles scientifiques aux Chapitres 4 et 5. Pour ce qui est du troisième objectif, les travaux d'un autre article scientifique sont exposés au Chapitre 6.

CHAPITRE 4

ARTICLE 1: ON IMPROVING THE PERFORMANCE OF LARGE PARALLEL LATTICE BOLTZMANN FLOW SIMULATIONS IN HETEROGENEOUS POROUS MEDIA

Article soumis en novembre 2008 pour publication dans *Computers & Fluids.*

On Improving the Performance of Large Parallel Lattice Boltzmann Flow Simulations in Heterogeneous Porous Media

David Vidal[1,2,*], Robert Roy[1] and François Bertrand[1,*]

[1]Ecole Polytechnique de Montréal, Montréal, H3C 3A7, QC, Canada

[2]FPInnovations - Paprican, Pointe-Claire, H9R 3J9, QC, Canada

david.vidal@fpinnovations.ca, {robert.roy, francois.bertrand}@polymtl.ca

4.0 ABSTRACT

Classical Cartesian domain decompositions for parallel lattice Boltzmann simulations of fluid flow through heterogeneous porous media are doomed to workload imbalance as the number of processors increases, thus leading to decreasing parallel performance. A one-lattice Lattice Boltzmann Method (LBM) implementation with vector data structure combined with even fluid node partitioning domain decomposition and fully-optimized data transfer layout is presented. It is found to provide nearly-optimal workload balance, lower memory usage and better computational performance than classical slice decomposition techniques using sparse matrix data structures. Predictive memory usage and parallel performance models are also established and observed to be in very good agreement with data corresponding to numerical fluid flow

* Corresponding authors.

simulations performed through 3-dimensional packings of cylinders and polydisperse spheres.

Keywords: Lattice Boltzmann method, fluid flow, porous media, SPMD parallelization, workload balance, memory usage

4.1 INTRODUCTION

The knowledge of transport phenomena in porous media is both of great scientific and technological importance. Nevertheless, porous media are among the most complex geometries that nature has to offer and are difficult to characterize. In this regard, fluid permeability is often used as it is a very sensitive material property that takes into account pore connectivity. Although empirical approximate correlations such as the Carman-Kozeny equation exist, the only theoretical way to precisely evaluate fluid permeability relies on the integration of the Navier-Stokes equations.

Conventional computational fluid dynamics (CFD) methods (i.e finite element/volume or finite difference methods) have proven limited in solving the Navier-Stokes equations in porous media. On the contrary, the Lattice Boltzmann Method (LBM), developed in the early 90s, is considered by many researchers as the method of choice for the simulation of single phase or multiphase flows in complex geometries [1,2]. In addition to its relative ease of implementation, LBM superiority is due to two main factors: 1) its flexibility in discretizing complex geometries by means of a simple structured lattice on which the fluid and solid phases are encoded in a Boolean manner, and 2) the inherent locality of its scheme, which makes it straightforwardly suitable for distributed parallelization.

Despite its advantages, three avenues of improvement have recently attracted the attention of researchers, in relation to the simulation of fluid flow in porous media: 1) the reduction of core memory usage, 2) the improvement of computational efficiency and accuracy of LBM algorithms, and 3) the reduction of workload imbalance associated with the heterogeneity of the domain when computing in parallel. In the case of memory requirement for porous media systems, several researchers [3-6] showed that transforming the sparse matrix data structure inherent to the LBM lattice into a vector data structure in which only the "fluid nodes" are retained (since no computations are performed on the "solid nodes") can significantly reduce memory consumption, especially when the domain porosity is lower than 70-75%. Martys & Hagedorn [6] and Argentini *et al.* [7] proposed simplifications of the LBM scheme in specific circumstances (for a LBM relaxation time equal to unity and for Stokes flows, respectively), which significantly reduce memory usage because, for each fluid node of the domain, they only require the storage of the fluid density and velocity or of a limited number of distribution moments. Along the same lines, Martys & Hagedorn [6] proposed a semi-direct addressing strategy based on the use of a pointer array, whereby memory allocation is required for the fluid nodes only.

The heuristic development of LBM has created a large variety of LBM schemes and implementations, and several researchers have tried to evaluate them. For example, Pan *et al.* [8] compared Bhatnagar-Gross-Krook single-relaxation-time (BGK) and multiple-relaxation-time (MRT) LBM schemes. They observed better accuracy with MRT schemes at the expense of a slightly higher computational cost (10-20%), although the BGK scheme can still provide accurate results when the single-relaxation-time parameter is equal to unity. Very recently, Mattila *et al.* [9] performed a comprehensive comparison in terms of computational efficiency and memory consumption of five different LBM implementations from the literature: well-established one-lattice two-step and two-lattice algorithms, and three recent implementations, namely the Lagrangian, shift (also called compressed-grid) and swap algorithms. They also investigated the effect of various data structures and memory addressing schemes on computational

efficiency. They found out that the newly developed swap algorithm [10] combined with a novel bundle data structure and indirect addressing yields both high computational performance and low memory consumption. The reader is referred to [9-12] and references therein for more thorough descriptions of these different implementations.

Concerning parallel efficiency, the classical Cartesian domain decompositions studied by Satofuka & Nishioka [13], which consist of dividing the whole domain in equally-sized subdomains (using slice- or box-decompositions), create a workload imbalance as the number of subdomains increases. As underlined by Pan *et al.* [5], this is due to porosity variations among the subdomains as their number is increased, not only for heterogeneous porous media but also for (macroscopically) homogeneous ones. To overcome this problem, several methods have been proposed such as the orthogonal recursive bisection [5,14] and multilevel recursive-bisection or k-way graph partitioning using the METIS package [3,4,15,16]. Although they may provide significant improvements with regard to classical domain decomposition techniques, these methods are uneasy to implement, come at high memory expense, create complex communication patterns and may become computationally expensive when dealing with very large systems (of say billions of lattice nodes) or when dynamic load balancing is required.

Recently, Wang *et al.* [17] proposed a quick, simple and elegant way to balance workload and reduce memory requirement for the two-lattice LBM implementation. The method consists of first vectorizing the data structure through the use of indirect memory addressing as previously proposed by others [3-5]. But to achieve accurate workload balance, the resulting data vector is then split into equally sized subvectors, each of which is assigned to a specific processor. As a result, exact fluid node load balance and high parallel efficiency can be achieved. Furthermore, a simple communication pattern among processors, similar to that with slice domain decomposition, is obtained since data communication for a given processor only involves the two nearest processors. Also, these authors claim that the data to be exchanged are contiguous in memory due to the vector data structure. However, it appears that the whole population set on lattice

nodes involved in ghost layers are exchanged between processors, which is more information than actually required. This can affect the data communication load and thus impair parallel performance. Moreover, despite the reduction of the memory requirements through the use of a vector data structure, we believe that further improvements could be achieved by resorting to LBM schemes that are more efficient than the two-lattice implementation.

The objective of this work is threefold: 1) to extend the parallel workload balancing procedure proposed by Wang *et al.* [17] to a more memory-efficient LBM algorithm, namely the one-lattice two-step LBM implementation, 2) to further improve the parallel performance of this scheme by reducing the communication overhead through the determination of a precise communication layout that minimizes the amount of data to be exchanged between the processors, and 3) to propose memory usage and parallel efficiency models that predict accurately and explain the performance of two one-lattice two-step LBM implementations, one with a sparse-matrix data structure and a classical slice domain decomposition, and one with a vector data structure and an even fluid node partitioning domain decomposition. First, the lattice Boltzmann method is recalled. There follows a description of the sparse-matrix and vector data structures used in the LBM implementations and their associated memory requirements. The parallel communication and workload balance strategies as well as their resulting performance are next examined. Finally, the computational advantages of the new methods proposed are assessed by simulating fluid flow through two different porous media made up of 3-dimensional packings of cylinders and polydisperse spheres, respectively.

4.2 THE LATTICE BOLTZMANN METHOD

Contrary to the conventional CFD methods that solve directly the Navier-Stokes equations, LBM actually "simulates" by means of a particulate approach the macroscopic behaviour of fluid molecules in motion. More precisely, LBM comes from the discretization in space (**x**), velocity (**e**) and time (t) of the Boltzmann equation from the kinetic gas theory, which describes the evolution of the probability distribution function (or population) of a particle, $f(\mathbf{x},\mathbf{e},t)$, and its microdynamic interactions.

4.2.1 Collision-propagation scheme

In practice, the populations of particles propagate and collide at every time step δ_t on a lattice with spacing δ_x and along $\mathbf{e}_i$ velocity directions, where the number of

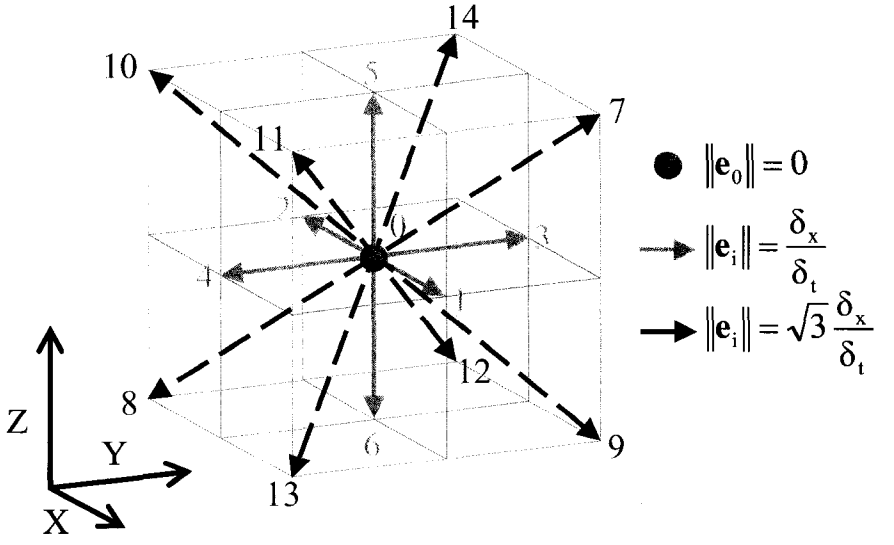


Figure 4.1 – Numbering of the 15 populations of the D3Q15 lattice used in the present work. Odd and even numbers correspond respectively to the forward-pointing and backward-pointing populations. Population 0 is a rest population.

directions i (n_d) depends on the type of lattice chosen. A D3Q15 lattice is used in the present work, i.e a 3-dimensional lattice with n_d=15 velocity directions (Figure 4.1). The collision-propagation procedure can be mathematically summarized by a two-step scheme encompassing a collision step,

$$f_i^*(\mathbf{x},t) = f_i(\mathbf{x},t) - \frac{f_i(\mathbf{x},t) - f_i^{eq}(\mathbf{x},t)}{\tau^*}, \tag{4.1}$$

followed by a propagation step,

$$f_i(\mathbf{x} + \mathbf{e}_i\delta_t, t + \delta_t) = f_i^*(\mathbf{x},t), \tag{4.2}$$

where $f_i(\mathbf{x},t)$ is the particle probability distribution function (or population) in the direction of the velocity $\mathbf{e}_i$ at position $\mathbf{x}$ and time t, and τ^* is a dimensionless relaxation time. The second term of the right-hand side of Equation (4.1) approximates the collision process by means of a single relaxation procedure, the so-called Bhatnager, Gross and Krook's approximation [18], towards a local equilibrium population that, for a D3Q15 lattice, is given by

$$f_i^{eq}(\mathbf{x},t) = w_i\rho\left[1 + 3(\mathbf{e}_i\cdot\mathbf{u})\left(\frac{\delta_t}{\delta_x}\right)^2 + \frac{9}{2}(\mathbf{e}_i\cdot\mathbf{u})^2\left(\frac{\delta_t}{\delta_x}\right)^4 - \frac{3}{2}(\mathbf{u}\cdot\mathbf{u})\left(\frac{\delta_t}{\delta_x}\right)^2\right], \tag{4.3}$$

with $w_0 = \frac{2}{9}$; $w_i = \frac{1}{9}$, for $i = 1:6$; $w_i = \frac{1}{72}$, for $i = 7:14$, where

$$\rho = \rho(\mathbf{x},t) = \sum_i f_i \tag{4.4}$$

and

$$\mathbf{u} = \mathbf{u}(\mathbf{x},t) = \frac{1}{\rho}\sum_i f_i\mathbf{e}_i \tag{4.5}$$

are the local fluid density and the local macroscopic fluid velocity, respectively. The dimensionless relaxation time τ^* is related to the kinematic viscosity ν of the fluid by

$$\tau^* = \frac{\nu}{\delta_t c_s^2} + \frac{1}{2}, \tag{4.6}$$

where c_s is a speed of sound defined as

$$c_s = \sqrt{\frac{3(1-w_0)}{7}} \frac{\delta_x}{\delta_t}. \tag{4.7}$$

In practice, for better accuracy, τ^* is chosen equal to 1, and δ_t and δ_x are chosen accordingly.

From initial conditions and appropriate boundary conditions, the collision-propagation scheme is marched in time until an appropriate convergence is reached (e.g. $(\mathbf{du}/dt)/(\mathbf{du}/dt)_{max}=10^{-5}$). The LBM scheme (Equations (4.1) & (4.2)) is explicit and the population update at a lattice node is a local operation since it only requires the populations of the immediate neighbouring nodes. This makes the LBM scheme well adapted to distributed parallelization. Note that, as previously mentioned, there are several ways to implement this scheme, which have recently been rigorously classified and studied by Mattila *et al.* [9]. In this work, we compare two one-lattice (two-step[39]) LBM implementations using two different data structures, as detailed in the Section 4.3.

4.2.2 Boundary conditions

In this work, three types of boundary conditions are used, which are typical for a porous medium. First, the boundary conditions at the periphery of the domain are

[39] Named two-step implementation by Mattila *et al.* [9] despite the fact that a two-step procedure can also be used within a two-lattice implementation.

periodic, which means that any out-going population re-enters the domain on its opposite side. For their implementation, a one-layer halo of nodes, called "periodic nodes", is added to the external boundaries of the domain to avoid breaking the pipeline of operations during the propagation step (see Figure 4.2(b)). Second, to impose a pressure drop ΔP in a given $\mathbf{e}_j$ direction, a body force is added on each node at each iteration in the $\mathbf{e}_i$ directions not normal to $\mathbf{e}_j$. Third, the wall boundary conditions on the solid objects of the porous domain can be modeled using the classical half-way bounce-back method, which reflects any in-coming population to the wall in the opposite direction at the next iteration. The solid boundary is in fact located halfway between the last fluid node and the first solid node when $\tau^*=1$. In practice, it is very convenient to tag as "bounce-back nodes" any solid nodes in direct connection with a fluid node (see Figure 4.2(b)), and

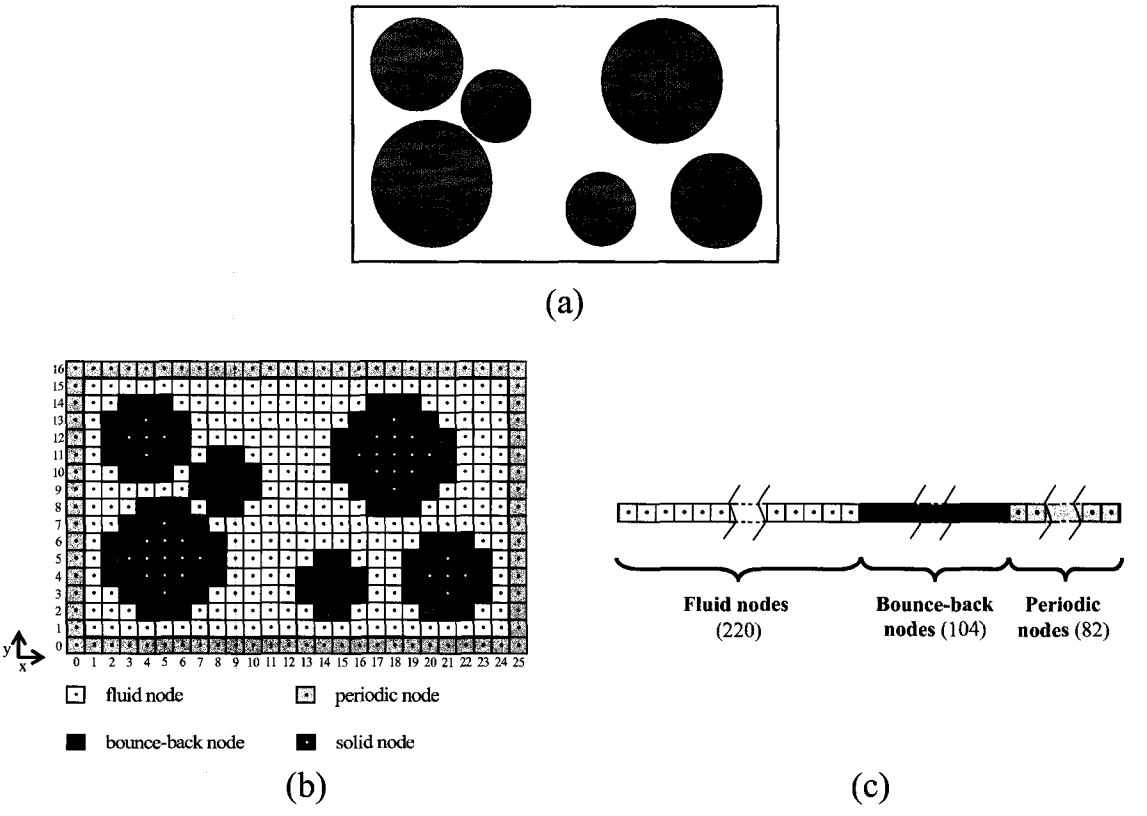


Figure 4.2 – 2D schematic discretization of a porous medium (a) using a 24×15 node domain. Here, the sparse matrix data structure (b) is compared with a (ordered) vector data structure (c). Note that the vector is made of three "sub-vectors" for storing population information related to fluid, bounce-back and periodic nodes, respectively.

copy and flip there any in-coming populations from the fluid nodes in provision of the following propagation step. For a more complete description of the boundary conditions and their implementation, as well as LBM in general, the reader is referred to [1,2].

4.3 DATA STRUCTURE AND MEMORY USAGE OPTIMIZATION

The original implementation of the LBM collision-propagation scheme, the so-called two-lattice algorithm, requires storing information on the n_d populations at the current and the next time steps in two 4-dimensional double precision arrays or matrices (three dimensions to locate the data in space and one dimension to identify the population). Using two matrices has the advantage of simplifying the algorithm and allows performing the collision and the propagation steps simultaneously (i.e. by fusing Equations (4.1) & (4.2)), although a two-step procedure is also feasible, but probably not as efficient. In addition, the geometry of the domain is stored in a 3-dimensional 1-byte array, or phase matrix, and allows retrieving the phase information, i.e. whether a node is a fluid or a solid node. In this case, the explicit tagging of periodic and bounce-back nodes is not required since it can be replaced by (expensive) if-conditions. Finally, additional space for the fluid density and the three components of the velocity can also be used, like in [17], but is not mandatory if the underlying calculations are performed locally as the collision step proceeds. For a $(n_x \times n_y \times n_z)$ lattice, the memory requirement (q_{mem}) for the two-lattice implementation with matrix data structure is then

$$q_{mem} \approx \{8 \text{ bytes}\} \times \underbrace{2 \times n_d \times (n_x \times n_y \times n_z)}_{\text{population storage}} + \{1 \text{ byte}\} \times \underbrace{(n_x \times n_y \times n_z)}_{\text{phase matrix}}, \tag{4.8}$$

where n_d is the number of populations used in the lattice, equal to 15 for a D3Q15 lattice, {8 bytes} indicates the computations are done in double precision and {x byte(s)} refers to a x-byte integer array for x<8.

It appears that the use of (tagged) periodic and bounce-back nodes and a clever order for the population updates during the propagation step may lead to a reduction of memory requirement, since only one 4-dimensional array or matrix is then necessary to store the population information required. For this to work, the order must avoid the overwriting of a yet to be propagated population by one that has been. This leads to the so-called one-lattice implementation that will be used hereafter. It can be shown that the memory requirement for the one-lattice implementation with matrix data structure is given by

$$q_{mem} \approx \{8 \text{ bytes}\} \times \underbrace{\left(n_d \times (n_x+2) \times (n_y+2) \times (n_z+2)\right)}_{\text{population storage}} + \{4 \text{ bytes}\} \times \underbrace{\left(n_b \times \left(4 \times 1 + 4 \times (n_d - 1)\right)\right)}_{\text{bounce-back treatment}} + \{1 \text{ byte}\} \times \underbrace{\left((n_x+2) \times (n_y+2) \times (n_z+2)\right)}_{\text{phase matrix}}, \tag{4.9}$$

where $n_b \approx \left(\alpha \frac{S}{\delta_x^2}\right)$ is the number of bounce-back nodes, which depends on the total surface area S of the solid phase, and $\alpha \geq 1$ a variable that depends on the shape of the interface and its orientation with respect to lattice reference axes (for cylinders aligned along one reference axis, we found $\alpha \approx 1.26$). The number of integer arrays required to store and treat the bounce-back nodes results from an algorithm proposed in [1]. On the downside, the presence of periodic and bounce-back node halos makes the adaptation of the parallel workload balance procedure proposed by Wang *et al.* [17] trickier. We will come back to this point in the next section.

Further memory gain can be made by considering that, for a porous domain of porosity ε, no operation takes place on solid nodes. In the above mentioned data structures, a great amount of memory is wasted in storing zeros for the solid nodes. As already mentioned, several researchers [3-5] have proposed to save up a significant amount of memory by linearizing into vectors the matrices and only storing information related to fluid nodes. As a result, the phase matrix is no more necessary. The downside is that the node topology provided by a matrix data structure is lost and direct addressing

between nodes is no more feasible. The vector data structure thus requires indirect addressing through the use of a connectivity list for all fluid nodes. Also, it is convenient to store the integer coordinate list of these nodes since such a list is required when post-processing the simulation results. The one-lattice implementation with vector data structure then entails the following memory requirement:

$$q_{mem} \approx \{8\text{ bytes}\}\times\underbrace{\left((n_f+n_p+n_b)\times n_d\right)}_{\text{population storage}}+\{4\text{ bytes}\}\times\Big(\underbrace{n_b\times\left(2\times(n_d-1)+1\right)}_{\text{bounce-back treatment}} +\underbrace{n_c\times(n_{d,in}+1+1)}_{\substack{\text{periodicity}\\ \text{treatment}}}+\underbrace{n_f\times(n_d-1)}_{\text{connectivity list}}\Big)+\{2\text{ bytes}\}\times\underbrace{\left(n_f\times 3\right)}_{\substack{\text{coordinate}\\ \text{list}}}, \tag{4.10}$$

where $n_c \approx \left((n_x+2)\times(n_y+2)\times(n_z+2)-n_x\times n_y\times n_z\right)\times\varepsilon$ is the number of periodic nodes, $n_f=\left(n_x\times n_y\times n_z\right)\times\varepsilon$ the number of fluid nodes, and $n_{d,in}$ the number of inward-pointing populations, which is equal to 5 for a D3Q15 lattice. Transforming the matrix data structure into a vector implies the replacement of the matrix of populations by three "sub-vectors", one for the fluid nodes, one for the bounce-back nodes and one for the periodic nodes, all three of which are combined into one single vector (as illustrated in Figure 4.2(c)). Similarly, for comparison purposes, the memory requirement for the two-lattice implementation with vector data structure is provided:

$$q_{mem} \approx \{8\text{ bytes}\}\times\underbrace{\left(2\times n_f\times n_d\right)}_{\text{population storage}}+\{4\text{ bytes}\}\times\Big(\underbrace{n_f\times(n_d-1)}_{\text{connectivity list}}\Big) +\{2\text{ bytes}\}\times\underbrace{\left(n_f\times 3\right)}_{\substack{\text{coordinate}\\ \text{list}}}. \tag{4.11}$$

The resulting memory usage for these different data structures will be assessed in Section 4.5, but it can be readily seen by comparing Equations (4.8)-(4.11) that a significant memory gain can be made if a vector is used instead of a matrix, since the population storage, which takes up most of the memory, is proportional to the porosity. In many applications, the porosity of the porous media is low.

Another issue related to the data structure is the ordering of the n_d populations in memory, which can affect the computational efficiency. When saving the populations in memory, one can either decide to store contiguously all the n_d populations of a given node and proceed nodewise, or store contiguously a given population for all the nodes and proceed populationwise. The first approach is called a collision-optimized data structure since it tends to give better performance during the collision step, whereas the second one is referred as the propagation-optimized data structure for opposite reasons. Mattila *et al.* [9] showed that the advantage of one over the other is processor dependent, with the propagation-optimized data structure giving better performance on Opteron processors, whereas the reverse is observed on Xeon processors. They also observed that a so-called bundle data structure, a somewhat hybrid version of these two structures, offer better overall performance on both processor types due to lower data cache misses. A propagation-optimized data structure was used in this work.

4.4 PARALLEL WORKLOAD BALANCE & COMMUNICATION STRATEGIES

Three key aspects of a parallel LBM implementation are examined in this section: computational workload, communication overhead and resulting parallel performance.

4.4.1 Workload balance strategy

With a vector data structure, it becomes straightforward to balance the workload on a parallel computer, as shown by Wang *et al.* in the case of a two-lattice LBM implementation without bounce-back and periodic nodes [17]. All that must be done is to ensure that each processor receives an equal portion of the vector data. In our one-lattice implementation, the situation is more complicated owing to the presence of periodic and bounce-back nodes. In theory, to achieve a perfect load balance, these periodic and bounce-back nodes would also need to be equally split between processors. In practice, this is hardly feasible, so that only the fluid node sub-vector is equally partitioned, as depicted in Figure 4.3(a). This could lead to some imbalance depending on the number of periodic and bounce-back nodes that each processor has to deal with. However, for some domain geometries, in the presence of large heterogeneities for instance, one could choose a preferential order of vectorization to limit the potential imbalance of periodic and bounce-back nodes. Nevertheless, the computational load required for the update of periodic and bounce-back nodes would still be very low when compared to the one required for the fluid nodes, so that no noticeable impact on the overall workload should be expected. Furthermore, note that no communication between processors related to the periodic or bounce-back nodes is required, so their impact on communication is null.

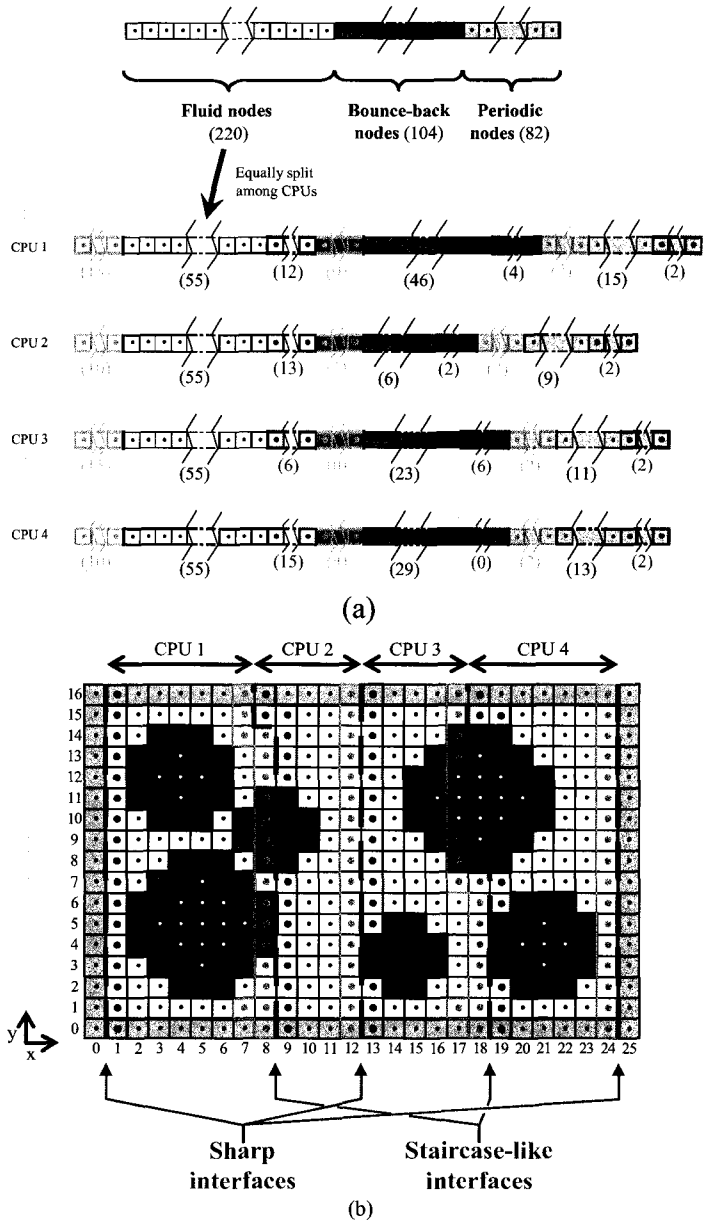


Figure 4.3 – Schematic splitting of the (ordered) vector of Figure 4.2 between 4 processors (CPUs) (a) and the resulting splitting on the original discretization (b). Note that the vectorization order (here y-x order) determines the location of the CPU interfaces (dashed red lines). Green and blue nodes are ghost layer nodes on the left and the right of each subdomain, respectively. No computations are performed on these ghost layer nodes. Only memory is allocated for the corresponding population data to be transferred during the propagation step.

4.4.2 Communication strategy

Overall, the workload balance procedure based on fluid nodes only leads to a slice domain decomposition method that resembles classical ones. For instance, y-x vectorizations in 2D and z-y-x vectorizations in 3D create slices in the x direction, as can been seen in Figures 4.3(b) & 4.4, respectively. In such cases, the data communication pattern between the processors is simple because each processor needs to communicate with only its two nearest processors. However, if not treated with care, the amount of data to be transferred between these processors can be substantially greater than in the case of classical slice decompositions. In fact, it depends on the position of the interface that can be sharp or staircase-like. In the best-case scenario, the sharp interface, only the 5 populations pointing towards the neighbouring processor and located in the last layer of nodes (for instance the layer I in Figure 4.4 and layer I and populations 1, 7, 9 ,11 &

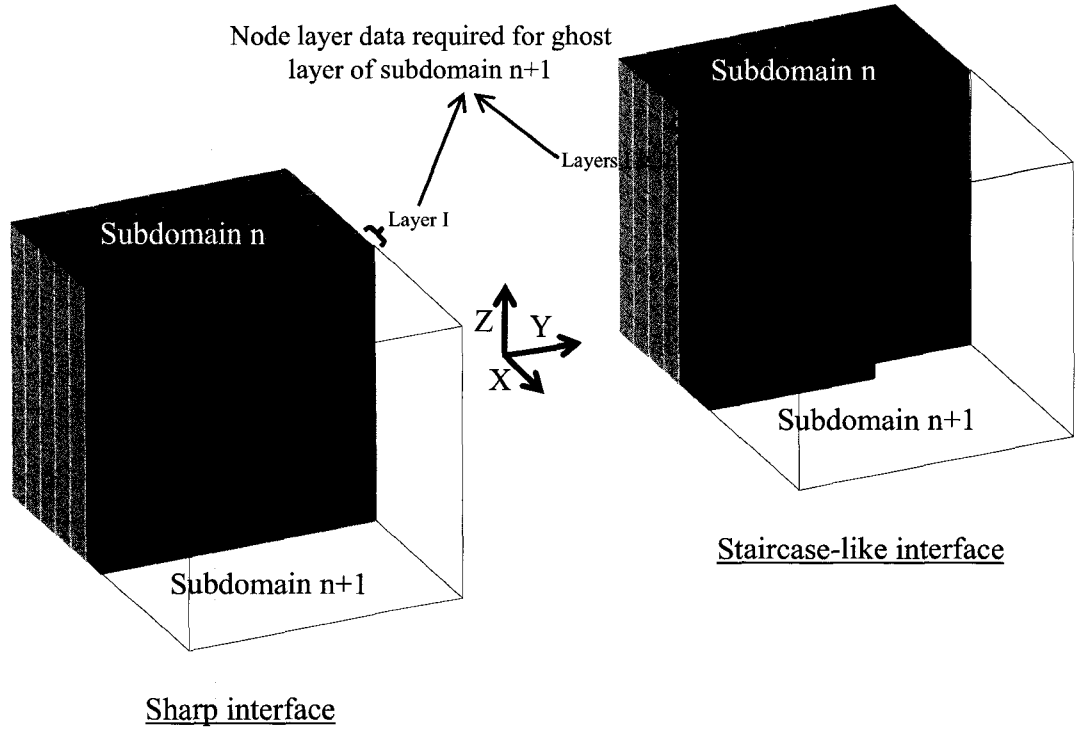


Figure 4.4 – Possible interface scenarios between subdomain n and subdomain n+1 (transparent) and the location of the layer data that need to be sent from processor n to processor n+1. Note that the vectorization order is here z-y-x.

13 in Figure 4.5, which both correspond to a x-forward data transfer) are required. In the worst-case scenario, the staircase-like interface, the 5 populations pointing towards the neighbouring processor are required but several of the 4 populations in the y-z plane of the interface (populations 3, 4, 5 & 6 for the x-forward direction) may also be required depending on the y-z position of the interface.

The easy solution is to send the 9 populations required (1, 3, 4, 5, 6, 7, 9, 11 & 13) to the neighbouring processor for all the fluid nodes of layers I & II (see the simple data transfer layout in Figure 4.5). This has the advantage of being straightforward to code, but entails increased communication overhead since more data than required are exchanged. Although a clear enough description of the data transfer layout they used is lacking, Wang *et al.* [17] seems to have adopted this strategy because they mention that the data to be exchanged are contiguous in memory. The actual amount (q_{com}) of data to be sent depends on the position of the interface in layer II. If the interface is located at the beginning of layer II (lower left corner) then $q_{com} \approx 9 \times (n_y \times n_z) \times \varepsilon$. In the worst case, if the interface is located at the end of layer II (upper right corner), $q_{com} \approx 2 \times 9 \times (n_y \times n_z) \times \varepsilon$. The likelihood to fall in the worst-case scenario at one interface increases with the number of processors used, and may yield a communication bottleneck. This scenario can be slightly improved at low implementation cost by using the so-called improved data transfer layout presented in Figure 4.5, wherein the amount of data to transfer is $q_{com} \approx 2 \times 7 \times (n_y \times n_z) \times \varepsilon$. Finally, a fully-optimized data transfer layout sending only the required population for each individual fluid node in layers I & II can be constructed (see Figure 4.5). As can be seen, the amount of communication is now equal to $q_{com} \approx 5 \times (n_y \times n_z) \times \varepsilon$, regardless of the location of the interface, which is equal to the amount of communication required by a sharp interface. In other words, such a layout guarantees a balanced communication workload between the processors. This comes at the cost of 1) a trickier to implement data transfer layout, which however needs to be established once for all at the pre-processing stage, and 2) a data preparation step at each iteration, since the data are no longer contiguous in memory. In this work, both the improved and the fully-optimized data transfer layouts are used and compared.

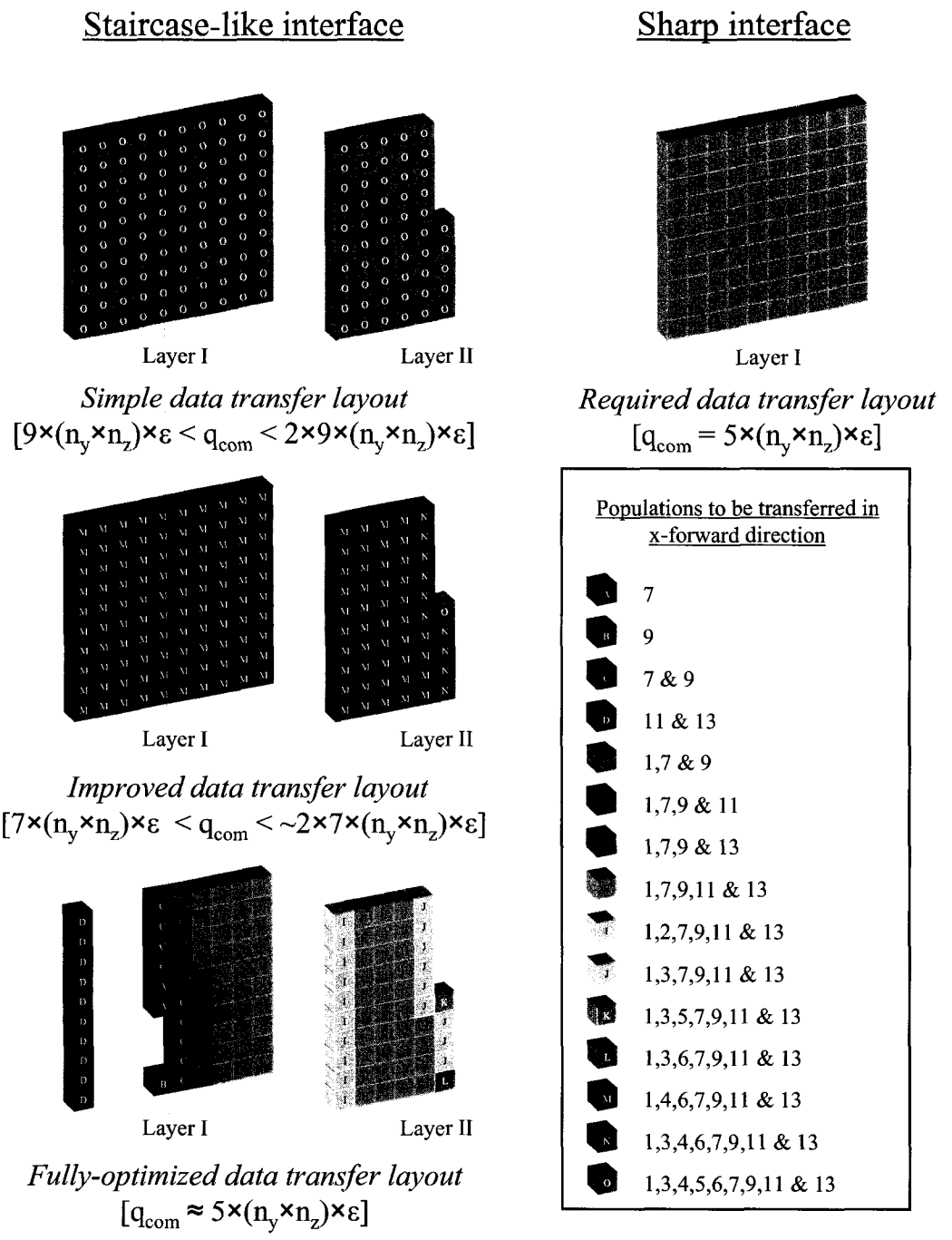


Figure 4.5 – Various population transfer layouts and resulting communication load (q_{com}) for the two types of interface (see Figure 4.4) and a D3Q15 lattice (see Figure 4.1) with periodic boundary conditions. Without loss of generality, assuming a z-y-x vectorization order, only the x-forward data transfer is presented. To construct the x-backward layout, opposite populations to the ones reported are used. Note that the data corresponding to solid, bounce-back and periodic nodes do not need to be exchanged. If the solid phase is static, the data transfer layout is determined once for all at the pre-processing stage.

4.4.3 Parallel performance

The parallelization of our LBM code was accomplished by means of a single-program multiple-data (SPMD) model, and implemented using MPI and a Fortran compiler. Communications between processors are carried out by non-blocking MPI_ISEND and MPI_IRECV subroutines. When evaluating parallel performance, one can either keep the lattice dimensions constant while increasing the number of processors (a speed-up scenario), or increase proportionally the lattice dimensions to the number of processors used (a scale-up scenario). If N_{tot} is the total number of lattice nodes and n_p the number of processors used for a given simulation, these two scenarios lead respectively to $N_{tot}=n_x \times n_y \times n_z$ and $N_{tot}=n_x \times n_y \times n_z \times n_p$. One can then derive for both scenarios the following theoretical efficiency $E(n_p)$ model for a porous media of average porosity ε (see Appendix):

$$E(n_p) = \frac{\varepsilon}{\varepsilon_{max} + \dfrac{q\, n_p \left(t_{lat} + s\, t_{data}\right)}{N_{tot}\, m\, t_{oper}}} = \frac{\varepsilon}{\varepsilon_{max} + r_{cc}} , \tag{4.12}$$

where ε_{max} is the highest subdomain porosity, m is the number of arithmetic (floating-point) operations per lattice node per iteration, q is the number of MPI_ISEND and MPI_IRECV required per processor (q=4 when each processor has two neighbours), t_{oper} is the average time spent per arithmetic operation, t_{lat} is the message latency, t_{data} is the average time to transfer 1 byte of data, s is the amount of data that needs to be transferred to one neighbouring processor, which is equal to $q_{com} \times 8$ bytes, and r_{cc} represents the ratio between the communication and the computational workload. The product $(m\, t_{oper})$ for a specific code on a given machine, heretoafter called the nodal computational time, can be approximated by

$$m\, t_{oper} \approx t_{CPU,1} / \left(n_{it} \times n_f\right) , \tag{4.13}$$

where $t_{CPU,1}$ is the CPU time measured on a single processor and n_{it} the number of iterations. The latency t_{lat} and data rate transfer t_{data} can be respectively evaluated using utilities such as *mpptest* [23] and *mpiP* [24], although only rough approximations can be obtained because the actual values depend on the number and the size of the messages transferred. Furthermore, the computational performance P_{comp} expressed in MLUPS (Millions of Lattice fluid node Updates Per Second) can be obtained by

$$P_{comp}(n_p) = \frac{10^{-6}\varepsilon N_{tot}}{\varepsilon_{max}\frac{N_{tot}}{n_p} m t_{oper} + q(t_{lat} + s t_{data})} = \frac{10^{-6} n_p}{m t_{oper}} E(n_p) \ . \tag{4.14}$$

Interestingly, this equation links the parallel efficiency to the computational performance, which can be evaluated experimentally as

$$P_{comp}(n_p) = \frac{10^{-6}\varepsilon N_{tot} n_{it}}{t_{CPU,n_p}} \ , \tag{4.15}$$

where t_{CPU,n_p} represents the CPU time with n_p processors. Combining Equations(4.14) & (4.15) gives a way to measure experimentally the parallel efficiency without timings on one single processor:

$$E(n_p) = \frac{\varepsilon N_{tot} n_{it} m t_{oper}}{n_p t_{CPU,n_p}} \ . \tag{4.16}$$

Note that simulations on one single processor may not be feasible because of the size of the computational domain. In fact, this equation allows to evaluate experimental parallel efficiencies for very large domains. It is valid if the computational time is proportional to the number of lattice nodes, which is the case for the LBM algorithms considered in this work.

From Equation (4.12), one can easily see that the efficiency is bounded by $\varepsilon/\varepsilon_{max}$ when communication overhead is negligible (i.e. $r_{cc} << \varepsilon_{max}$), which means that, in such a

case, any porosity variation among subdomains will significantly affect the parallel performance. Also, when the number of subdomains increases, the likelihood of an increase of the subdomain porosity variability grows rapidly, all the more so when the pore features are in the same order of magnitude as the subdomain dimensions, even for apparently homogeneous structures, as pointed out in [5] and illustrated by Figure 4.6 in the case of Fontainebleau sandstone. Data presented in this figure show that if 64 processors were used to compute fluid flow through a ~5 mm^3 Fontainebleau sandstone discretization, a parallel efficiency as low as 41.4% (=15.0%/36.5%) would at best be observed. If communication overhead is not an issue, the one-lattice implementation with vector data structure should provide instead a theoretical parallel efficiency of 100% because $\varepsilon_{max}= \varepsilon$ and workload is balanced.

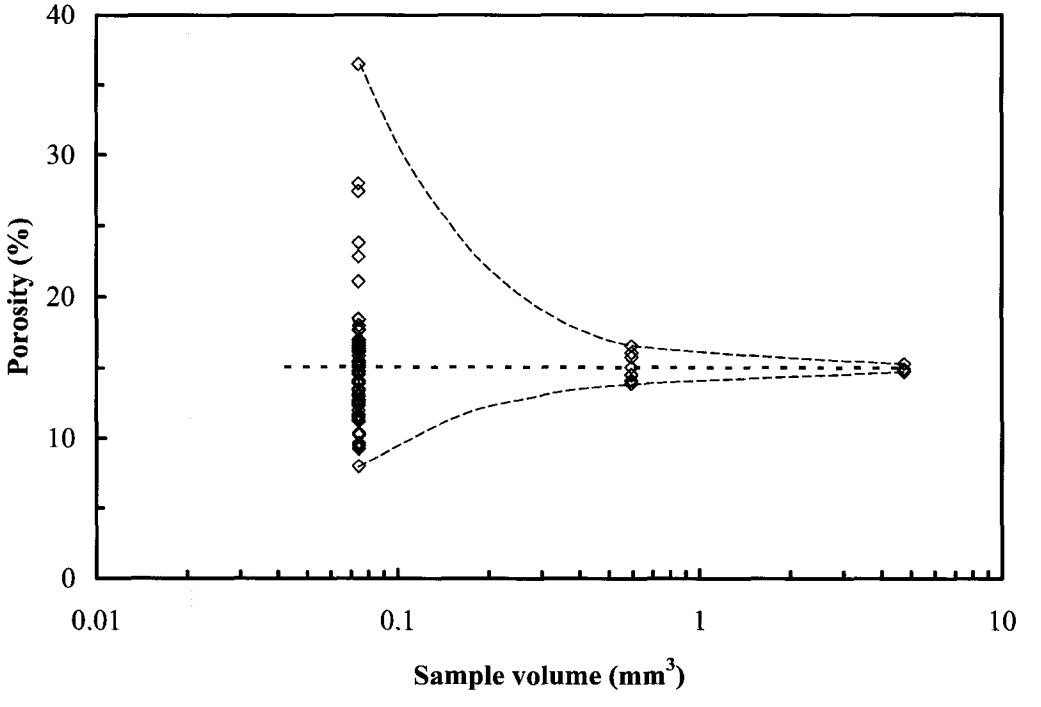


Figure 4.6 – Experimental porosity variations of a Fontainebleau sandstone as a function of sample volume (drawn from X-ray microtomography data published in [22]). The volume ratios between the three samples and the largest one are 1, 1/8 and 1/64.

4.5 NUMERICAL EXPERIMENTS

In order to demonstrate the efficiency of the proposed algorithms, both 3-dimensional fluid flow simulations through a hexagonal packing of cylinders and a heterogeneous random packing of polydisperse spheres were conducted and compared with those obtained using classical slice domain decomposition.

4.5.1 Hexagonal packing of cylinders

The following numerical simulations were performed on the High-Performance Computing (HPC) cluster (Mammouth(mp)) from the Réseau Québécois de Calcul de Haute Performance (RQCHP). Table 4.1 summarizes the main features of this HPC cluster. The dimension of the lattice was increased proportionally to the number of processors used (scale-up test). More precisely, the total number of lattice nodes (N_{tot}) was $600 \times 10\ n_p \times 347$ lattice nodes. Four different cylinder radii, R=60.6, 67.1, 73.6 and 80.1 lattice nodes, were considered to investigate the impact of porosity variations on performance. An example of a computed flow field is presented in Figure 4.7. The comparison between the numerical and analytical values of fluid permeability given in Figure 4.8 shows the accuracy of our LBM code. Table 4.2 presents the nodal computational times ($m\ t_{oper}$) of the various codes used on Mammouth(mp), as calculated from Equation (4.13).

Table 4.1 – Specifications of the Mammouth(mp) HPC cluster.

	Mammouth(mp) parallel cluster
Make	Dell PowerEdge SC1425
Processors used	2× 128 Intel Xeon[†]
- clock speed	3.6 GHz
- bus FSB	800 MHz
- RAM	8 GB
- cache	1 MB L2
Interconnection	Infiniband 4x (700-800 MB/s)
- t_{lat} (μs)	~5.0
- t_{data} (ns/byte)	~8.0
Operating system	RedHat Enterprise Linux 4 (2.6.9-42.0.3.ELsmp)
Compiler	Portland Group pgf90 Fortran (6.0-4)
Message passing	MPI (mvapich2 0.9.82)

[†] Only one processor per server was used.

Table 4.2 – Nodal computational time (m t_{oper}) for the various codes on the Mammouth(mp) HPC cluster.

Code	Nodal computational time (ns) with 4-byte integer compiler option	with 8-byte integer compiler option
One-lattice algorithm with vector data structure	879	N/A
One-lattice algorithm with sparse-matrix data structure	1142	N/A

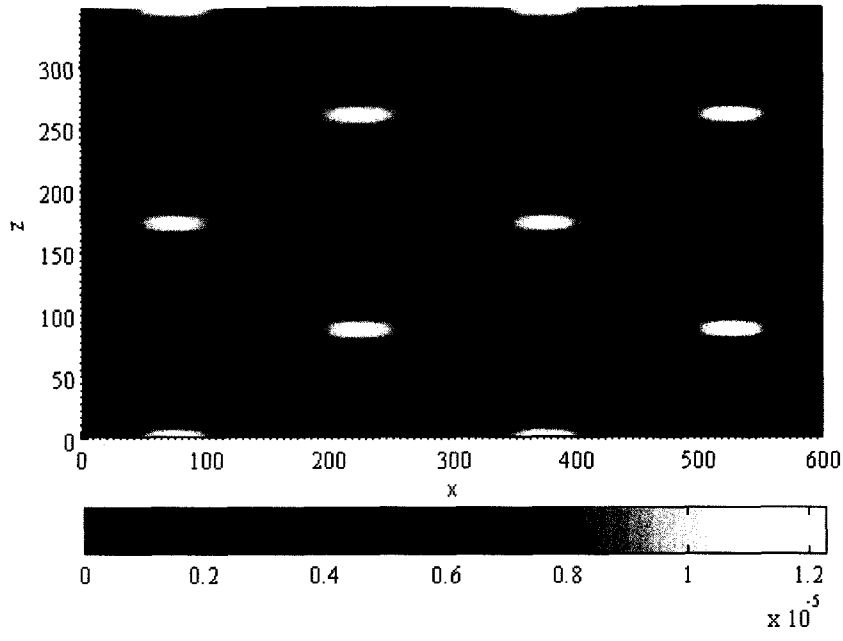


Figure 4.7 – Y-cross section of a hexagonal packing of cylinders and the resulting flow velocity as computed by LBM (blue color chart). The cylinder radius R is 73.6 lattice nodes (δ_x = 0.0462 μm), which results in a domain porosity of 34.5%. Note that the pressure drop is imposed in the x direction.

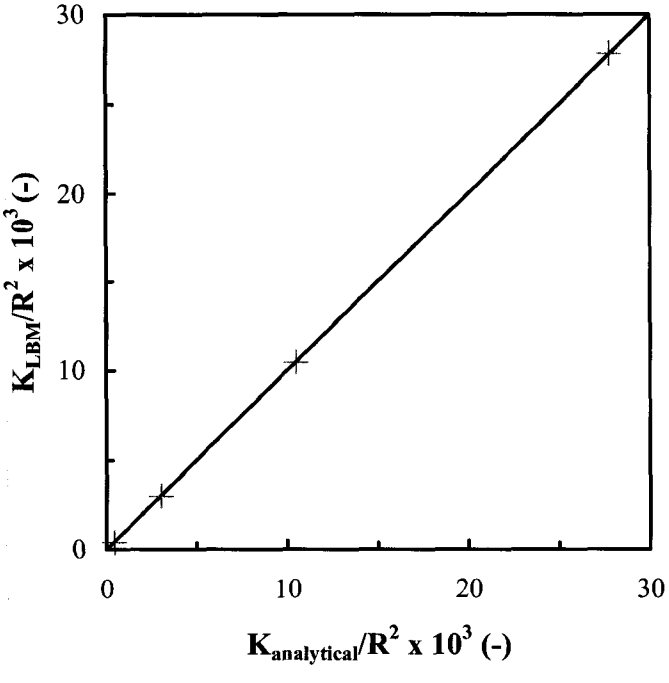


Figure 4.8 – Comparison between the x-axis normalized fluid permeability predicted by LBM and the analytical solution [23] for hexagonal packings of cylinders with four different radii R. The relative errors are smaller than 1.4%.

The memory requirements predicted by Equations (4.8)-(4.11) show that the vector data structure offers a significant improvement over the sparse matrix data structure for both two-lattice and one-lattice implementations, when $\varepsilon \leq 80\%$ and $\varepsilon \leq 70\%$, respectively (see Figure 4.9). This is in good agreement with previously reported data [3-5]. At high porosity values, the small memory gain resulting from the removal of the solid nodes is surpassed by the cost of the coordinate and connectivity lists. As can also be seen for the one-lattice implementations, the finer the lattice size the lower the memory requirements per lattice node since a smaller proportion of bounce-back nodes are needed to discretize the interface between the fluid and solid phases. In this regard,

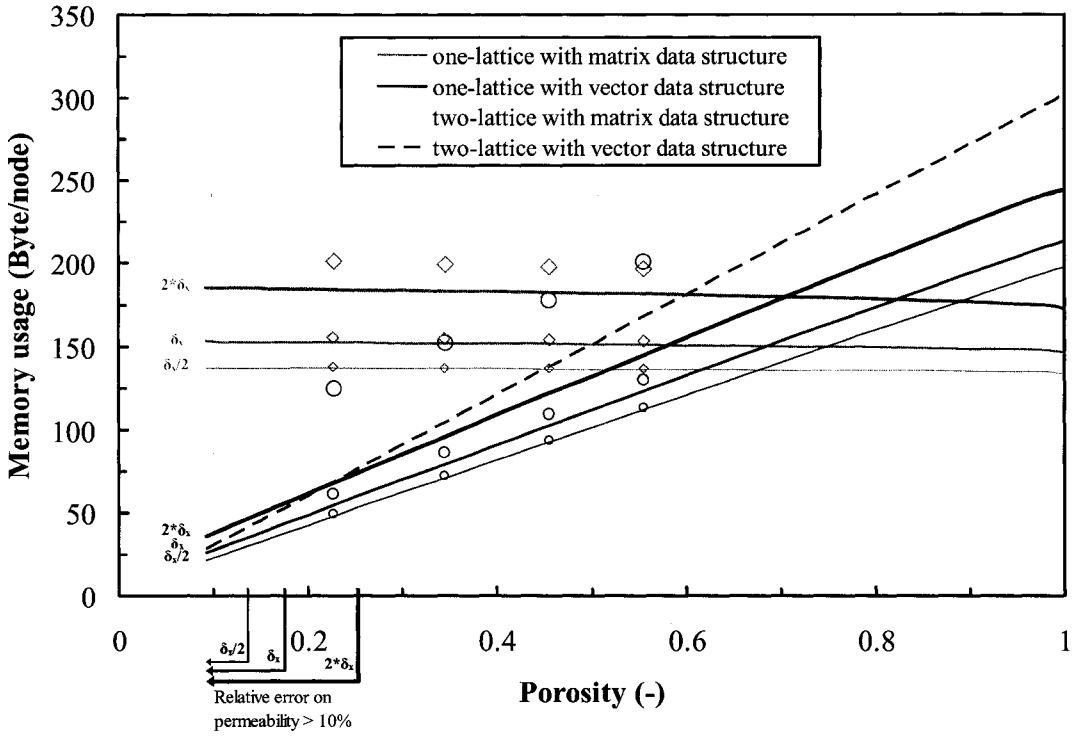


Figure 4.9 – Memory usage per node as a function of the porosity of a hexagonal packing of cylinders for one- and two-lattice LBM implementations with sparse matrix and vector data structures. The porosity is changed by varying the diameter of the cylinders. Lines correspond to model predictions (Equations (4.8)-(4.11)) and symbols are actual numerical experiment data points for the one-lattice implementations only. The thicker the lines or the bigger the symbols, the coarser the lattice (δ_x = 0.0462 μm). The porosity values for the three lattice resolutions, below which the computed permeabilities are off by more than 10% from the analytical solution, are displayed on the left of the x-axis for the three lattice resolutions.

the actual memory requirements for both one-lattice implementations are in good agreement with the predictions for the finest lattices. For the coarse lattices, a deviation from the predictions is observed. It appears that, due to the small size of the domains involved in these test cases, additional secondary arrays not taken into account in the memory models are no longer negligible. Finally, according to the theoretical models, the one-lattice implementation with vector data structure provides a noticeable reduction of memory usage compared to its two-lattice counterpart, especially for fine lattices. For coarse lattices and low porosity (< ~20%), the corresponding two-lattice implementation is less greedy although the lattice resolution is such that poor accuracy is expected with a relative error on permeability larger than 10%. Furthermore, note that the two-lattice implementations of Wang *et al.* [17] require significantly more memory (~40%) than the values predicted here since they also store the density and the three components of the velocity, and use a D3Q19 lattice, i.e. 19 populations instead of 15 in the case of the D3Q15.

For this simple case study, there is an obvious slice domain decomposition along the y direction that can lead to high parallel efficiency due to straightforward workload balance and constant communication load (related to the exchange of data for 600×347 lattice nodes). In this regard, with R=73.6 lattice nodes and $2 \leq n_p \leq 100$, nearly constant parallel efficiencies of 96.7%, 96.8% and 98.8% were obtained, respectively, for the one-lattice implementation with x-slice decomposition, and for the one-lattice implementations with even vector partitioning using improved and fully-optimized data transfer layouts.

To highlight performance improvements with even vector partitioning domain decomposition when load imbalance is present, the computational domain in Figure 4.7 was discretized and purposely decomposed in the x direction (i.e. the domain was swept during vectorization following the z-y-x order). In the case of an x-slice classical decomposition, the porosity ratio $\varepsilon/\varepsilon_{max}$ can be evaluated analytically as a function of the number of subdomains and showed to decrease to a value as low as ε for $n_p \geq 75$,

because $\varepsilon_{max} = 100\%$ in such cases. As expected and as can be seen in Figure 4.10*(a)*, the resulting load imbalance creates a significant drop in parallel efficiency, which reaches values as low as near ε=34.5%. This experimental drop in parallel efficiency is in fact very well predicted by Equation (4.12) (*green dashed line* in Figure 4.10*(a)*). A substantial gain in performance as well as the elimination of the fluctuations due to porosity variations among processors can be observed for the proposed one-lattice implementations with even vector decomposition. The decrease in performance is now solely due to the increase in message passing as the number of processors is increased (we recall that $s \propto n_p$ for an x-axis decomposition). The imbalance in the number of periodic and bounce-back nodes from one processor to another does not seem to affect performance as this would create noticeable fluctuations, which justifies our approach consisting in only balancing the number of fluid nodes. At 100 processors, 75% efficiency is achieved with the fully-optimized data transfer layout, which is much better than 35% with the x-axis decomposition, and significantly better than ~60% with the improved data transfer layout. As expected, the performance obtained with the fully-optimized data transfer layout surpasses the one achieved with the improved data transfer layout. The difference is well predicted by the performance model (*red* and *orange dashed* lines, respectively, obtained from Equation (4.12) with $\varepsilon_{max} = \varepsilon$). Interestingly, as can be seen in Figure 4.10*(b)*, the sequential execution of the one-lattice codes with even vector partitioning is ~1.3 times faster than their counterpart with classical slice domain decomposition. This can be attributed to the removal of expensive if-conditions to test the state of the nodes when scanning through the sparse matrix data structure. Higher speed ratios are achieved as the number of processors is increased, to reach a maximum of ~2.7 at 100 processors.

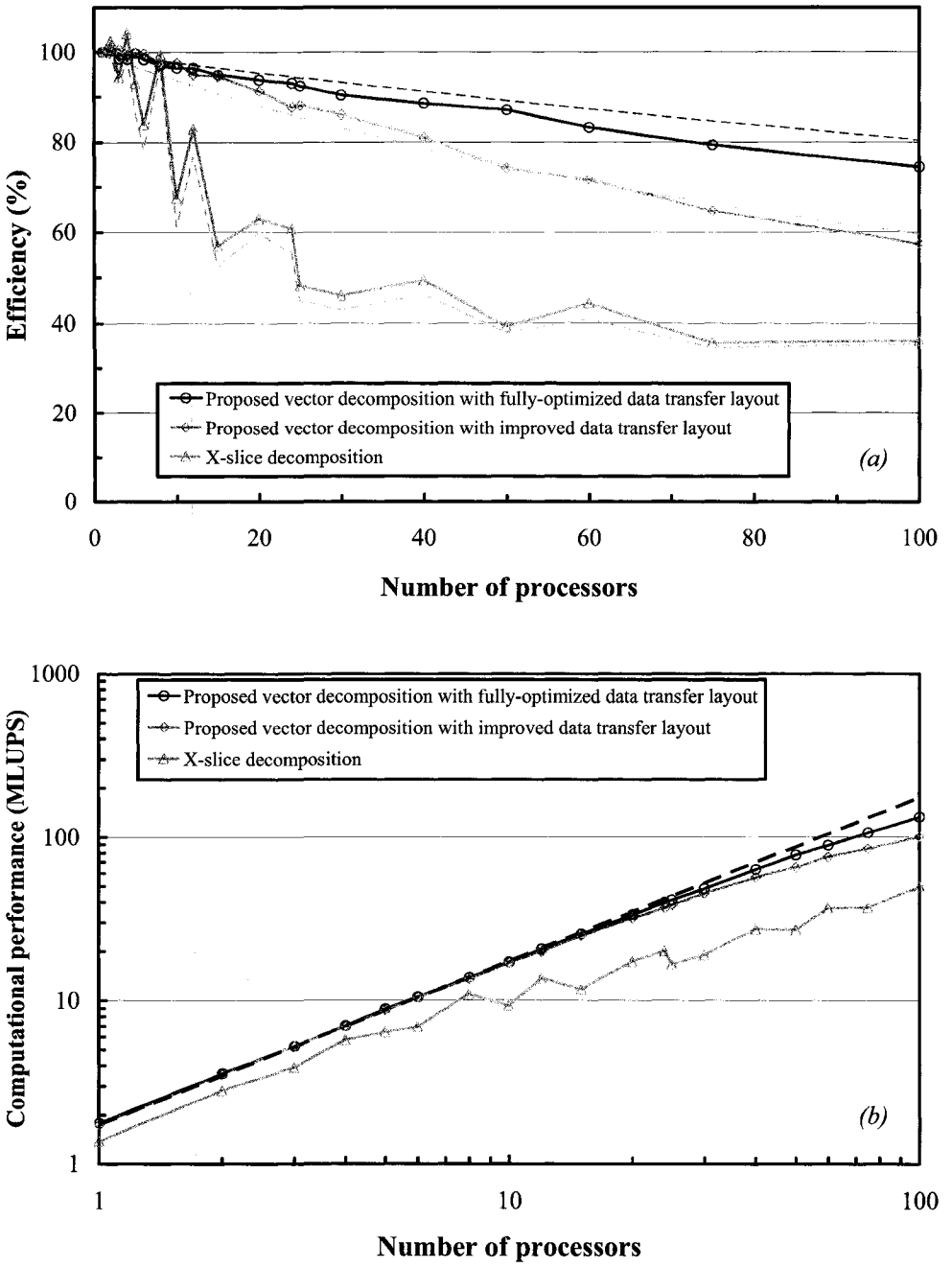


Figure 4.10 – Parallel efficiency *(a)* and computational performance *(b)* comparisons on the Mammouth(mp) cluster between x-slice domain decomposition and even vector partitioning domain decomposition with both improved and fully-optimized data transfer layouts for a hexagonal packing of cylinders (R=73.6 lattice nodes) and proportional domain size (scale-up test). The *colored dashed lines* in *(a)* represent the model predictions from Equation (4.12) for the corresponding domain decompositions. In *(b)*, the *black dashed line* represents the theoretical linear performance for the even vector partitioning domain decomposition.

4.5.2 Random packing of polydisperse spheres

To further investigate the parallel performance of the proposed LBM implementations, a second series of tests was carried out on a 3-dimensional random packing of polydisperse spheres generated using a Monte-Carlo packing procedure described elsewhere [24] (Figure 4.11). To induce large scale heterogeneities within the packing, 6 ellipsoidal pore inclusions were introduced randomly within the packing. The domain size (400^3 lattices nodes), which could fit in the memory of a single server, was kept constant as the number of processors was increased (speed-up test). Note that, for this case, the memory usage on a single processor using the vector data structure was 42% lower than that for the sparse matrix data structure (5.9 GB vs. 10.1 GB), which represents a substantial reduction. The tests were performed on the HPC cluster (Artemis) from FPInnovations. Table 4.3 summarizes the main features of this HPC cluster and Table 4.4 presents the nodal computational times (m t_{oper}) of the various

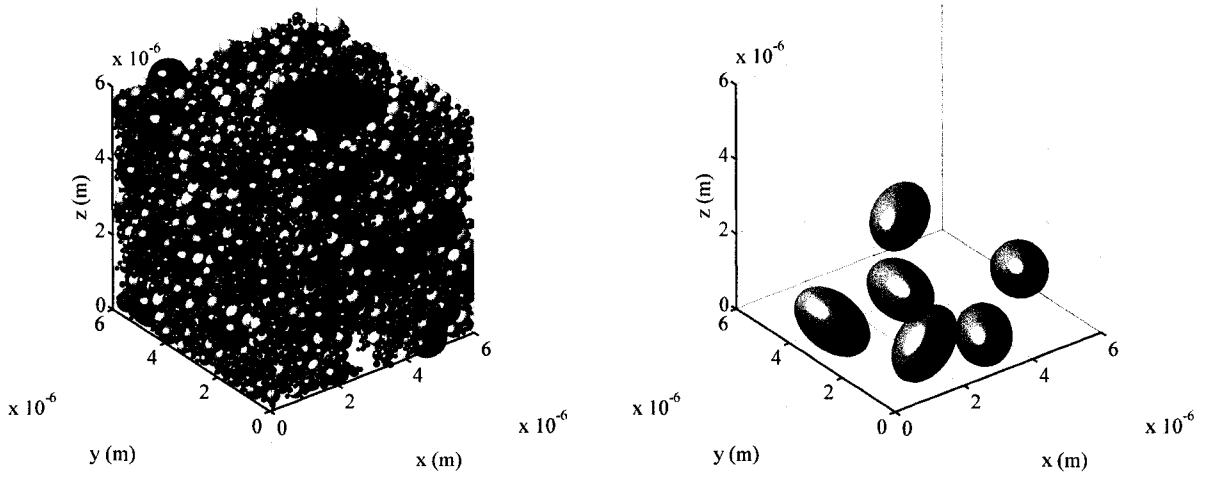


Figure 4.11 – Random packing with a lognormal particle size distribution (geometric standard deviation equal to 2.5 and median particle size of 0.6 μm), as created using a Monte-Carlo packing procedure described in [24] (*left*). 45 different particle sizes were used ranging from 0.1 μm to 4 μm. The warmer the particle color, the smaller its size. Ellipsoid-shaped pore inclusions (total volume equal to 6×~1.85 μm^3 and ellipsoid aspect ratio equal to 1.5) were introduced within the packing to induce large scale heterogeneities (*right*). The overall packing porosity is equal to 27.5%.

codes used on this cluster.

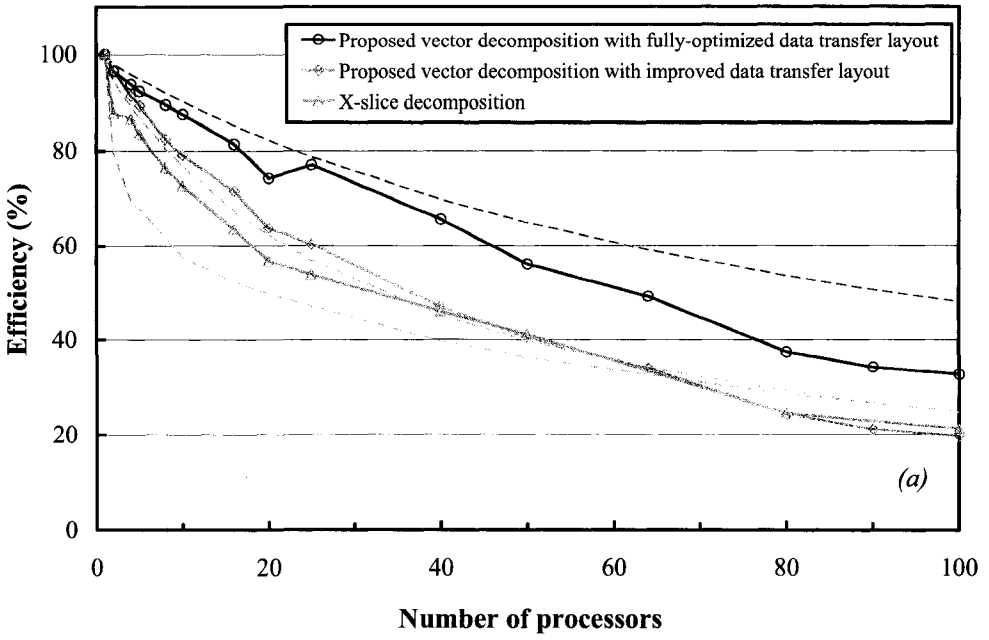
Table 4.3 – Specifications of the Artemis HPC cluster.

	Artemis parallel cluster
Make	Dell PowerEdge 1950
Processors used	2× 64 quad core Intel Xeon 5440
- clock speed	2.83 GHz
- bus FSB	1333 MHz
- RAM	16 GB
- cache	12 MB L2
Interconnection	Gigabit Ethernet
- t_{lat} (μs)	~1.0
- t_{data} (ns/byte)	~12.0
Operating system	CentOS 4.6 (2.6.9-67.0.15.ELsmp)
Compiler	Intel Fortran (10.1.015)
Message passing	MPI (OpenMPI 1.2.6)

Table 4.4 – Nodal computational time (m t_{oper}) for the various codes on the Artemis HPC cluster.

Code	Nodal computational time (ns) with 4-byte integer compiler option	with 8-byte integer compiler option
One-lattice algorithm with vector data structure	443	478
One-lattice algorithm with sparse-matrix data structure	704	802

A significant decrease in parallel performance can be observed in Figure 4.12 as the number of processors increases for the three algorithms investigated. This is due to the reduction of the subdomain granularity as the number of processors increases, which leads to an increase of the communication over computation ratio, r_{cc} in Equation (4.12). As the weight of r_{cc} becomes more and more important than that of ε_{max} in this equation,



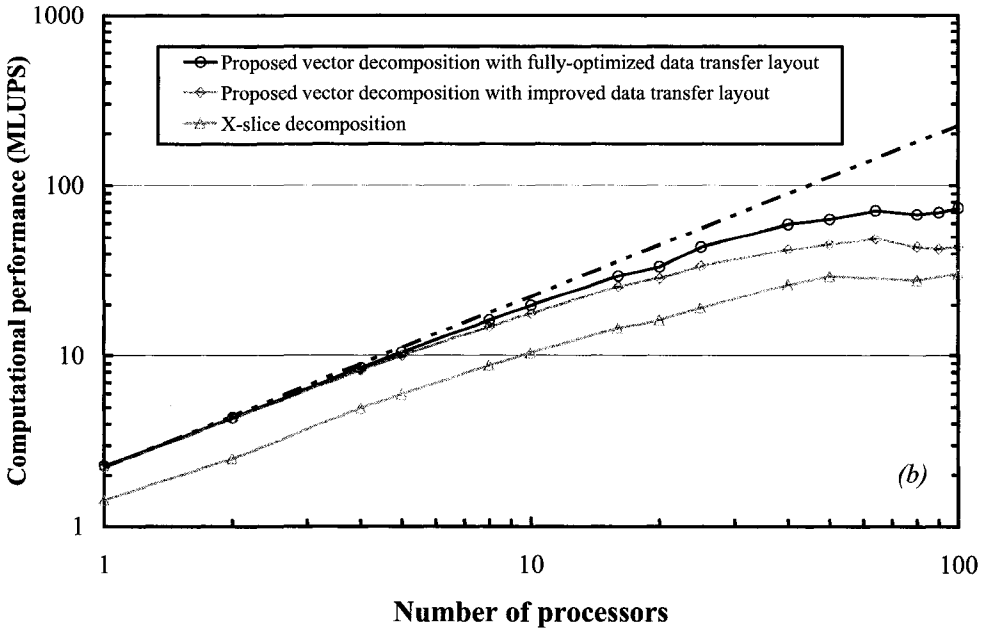


Figure 4.12 – Parallel efficiency *(a)* and computational performance comparisons *(b)* on the Artemis cluster between x-slice domain decomposition and even vector partitioning domain decompositions with both improved and fully-optimized data transfer layouts, for a random packing of spheres with constant domain size (speed-up test). The *colored dashed lines* in *(a)* represent the model predictions from Equation (4.12) for the corresponding domain decompositions. In *(b)*, the *black dashed line* represents the theoretical linear performance for the even vector partitioning domain decomposition.

the upper bound for the parallel efficiency gradually switches from $\varepsilon/\varepsilon_{max}$ to ε/r_{cc}. In other words, this means that the communication overhead becomes more important than the workload imbalance with regard to the parallel performance. Moreover, when the number of processors is small, the use of the one-lattice vector implementation with improved data transfer layout (with $s \propto 14\times n_y\times n_z\times\varepsilon$) yields a slight improvement of the parallel efficiency with respect to the one-lattice sparse matrix implementation (with $s \propto 5\times n_y\times n_z$), which results from a smaller amount of transferred data ($14\varepsilon \sim 3.9 < 5$). When the number of processors increases, the efficiencies become similar. However, it can be noted in Figure 4.12*(b)* that the former remains ~1.5 times faster than the latter over the whole range of number of processors investigated. Furthermore, thanks to its careful data transfer layout, the one-lattice vector implementation with fully-optimized layout (with $s \propto 5\times n_y\times n_z\times\varepsilon$) provides a noticeable improvement over the other two algorithms. These trends are also confirmed by the performance model predictions.

To assess the parallel performance as the domain size is increased (scale-up test from 384^3 to 1792^3 node lattices),simulations were carried out with 128 processors on HPC cluster Artemis for all three algorithms. The computational performance and parallel efficiency was calculated through Equations (4.15) & (4.16) for the three algorithms (Figure 4.13). As expected, the computational performance and efficiency increase with the number of lattice (fluid) nodes in all cases. Moreover, the vector decomposition with the improved data transfer layout provided enhanced performance as compared to the sparse matrix implementation with classical slice domain decomposition. This is mainly due to the better sequential performance of the resulting code as the efficiency is only slightly better. On the other hand, resorting to the fully-optimized data transfer layout outperformed the other two implementations both in terms of efficiency and performance. The model predictions from Equations (4.12) & (4.14) (*colored dotted lines*) become extremely good above $\sim 1.2\times10^8$ fluid nodes (768^3 node lattices). As all algorithms follow closely the performance and efficiency models, it can be concluded that they converge to the expected bounds, that is $E\rightarrow 1$ (balanced workload) for the vector decompositions and $E\rightarrow\varepsilon/\varepsilon_{max}$=27.5%/42.2%=65.2%

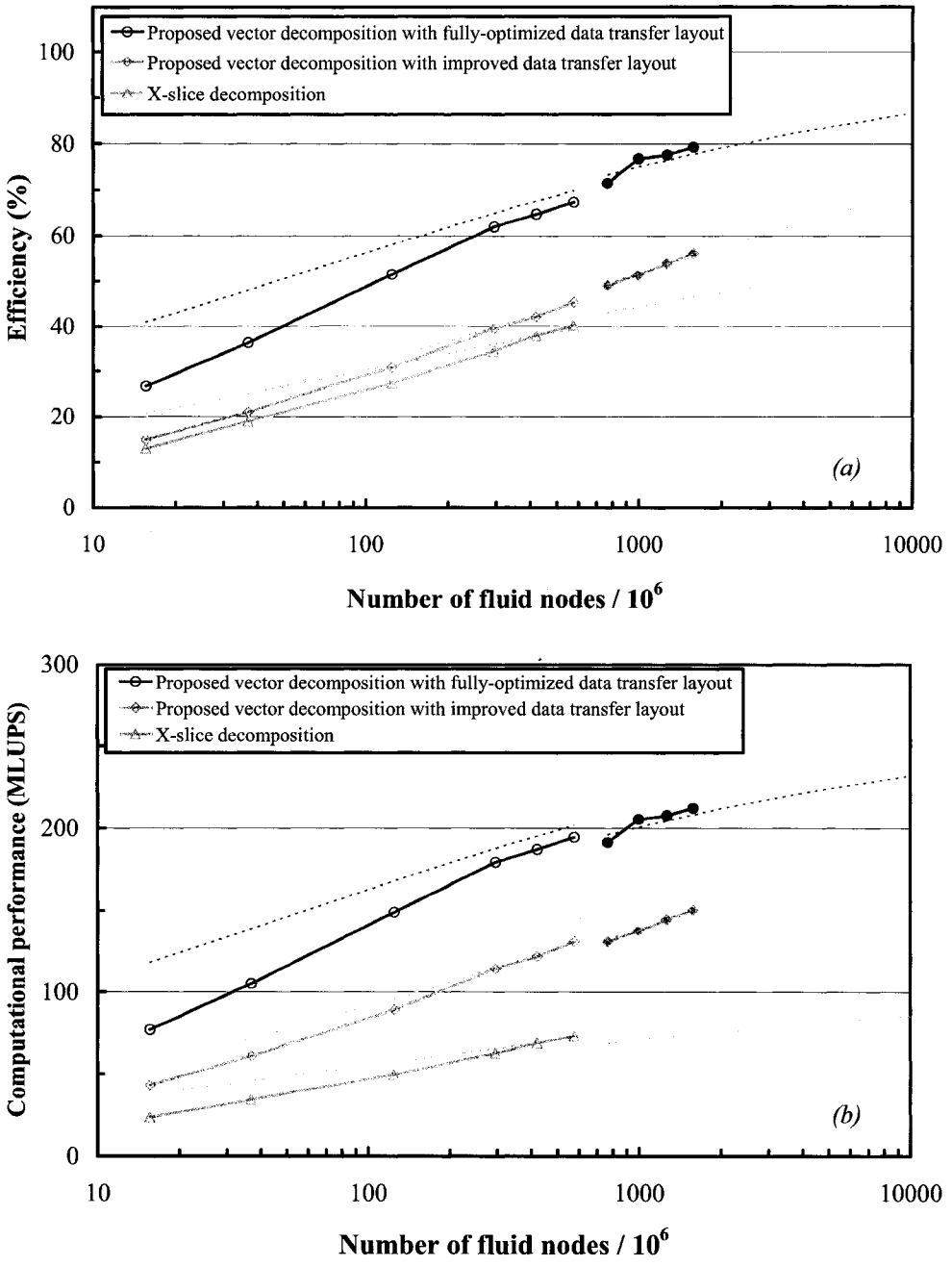


Figure 4.13 – Parallel efficiency (*a*) and computational performance *(b)* comparisons at 128 processors on the Artemis cluster between x-slice domain decomposition and even vector partitioning domain decompositions with both improved and fully-optimized data transfer layouts, for a random packing of spheres with constant domain size (scale-up test). The *colored dotted lines* represent the model predictions from Equations (4.12) & (4.14) for the corresponding domain decompositions. Open and filled symbols correspond to simulations performed respectively with 4- and 8-byte integers.

(unbalanced workload) for the slice domain decomposition. Also note the small decrease in performance when simulations were performed with 8-byte integers (*closed symbols*), which were required whenever the number of lattice nodes was too large to be indexed by 4-byte integers. Finally, the fact that domains three times larger (comprising up to 1.6 billions fluid nodes) could be simulated in the case of even vector partitioning domain decompositions highlights the memory advantage of this strategy over the use of sparse matrices.

4.6 CONCLUSION AND PERSPECTIVES

A one-lattice vector implementation of LBM with even fluid node partitioning domain decomposition was introduced and shown to 1) substantially reduce the memory usage when domain porosity is low (below ~70%), 2) decrease sequential execution time thanks to the removal of costly if-conditions to test the state of the nodes when scanning through the sparse matrix data structure, and 3) eliminate the workload imbalance resulting from local heterogeneities of a porous structure and thus improve parallel performance. It was found that although workload balancing is only carried out on the fluid nodes, the imbalance in bounce-back and periodic nodes does not affect measurably the performance. Consequently, this indicates that the algorithm is nearly-optimal for load balancing. Although the communication patterns are simple (i.e. communications involving only two neighbouring processors like in slice domain decomposition), it was observed that the amount of populations to be transferred to the neighbouring processors should be minimized to reduce communication overhead. Consequently, a fully-optimized data transfer layout for the D3Q15 lattice and periodic boundary conditions, easily adaptable to other types of lattice, was developed and showed to outperform other vector and matrix data transfer layouts. Predictive memory usage and parallel

performance models were also established and found to be in very good agreement with the measured data.

The proposed method still retains one of the drawbacks from classical slice domain decompositions when dealing with large interfaces between subdomains combined with small problem granularity, which can create an important communication overhead. We believe that parallel performance in such cases can be better with methods that minimize subdomain interfaces such as the spectral recursive bisection and multilevel graph partitioning methods. However, this drawback could be alleviated by simplifying the LBM algorithm and further reducing the amount of data to be exchanged. This will be the subject of a forthcoming paper. Finally, the proposed method appears to be well-suited for the (dynamic) load balancing of simulations of fluid flow through very large and complex heterogeneous porous domains such as those involving settling particles [25] or packings of highly polydisperse spheres [24]. It is likely to run efficiently on heterogeneous architectures because the fluid node vector can be judiciously split according to the relative computational speed of the processors involved in a parallel simulation.

4.7 ACKNOWLEDGMENTS

The computer resources and support from the Réseau Québécois de Calcul de Haute Performance (RQCHP) and from FPInnovations, as well as the financial contribution of the NSERC Sentinel Network are gratefully acknowledged. Special thanks to Louis-Alexandre Leclaire and Christian Poirier.

4.8 REFERENCES

[1] Succi S. The lattice Boltzmann Equation for Fluid Dynamics and Beyond. Oxford, UK: Oxford Science Publications; 2001

[2] Nourgaliev RR, Dinh TN, Theofanous TG, Joseph D. The lattice Boltzmann equation method: theoretical interpretation, numerics and implications. Int J Multiphase Flow 2003; 29(1):117-69

[3] Dupuis A, Chopard B. An object oriented approach to lattice gas modeling. Future Gener Comput Syst 2000;16(5):523-32

[4] Schulz M, Krafczyk M, Tolke J, Rank E. Parallelization strategies and efficiency of CFD computations in complex geometries using lattice Boltzmann methods on high performance computers. In: Breuer M, Durst F, Zenger C, editors. High performance scientific and engineering computing. Berlin: Springer Verlag; 2002. p.115-22

[5] Pan C, Prins JF, Miller CT. A high-performance lattice Boltzmann implementation to model flow in porous media. Comput Phys Commun 2004; 158:89-105

[6] Martys NS, Hagedorn JG. Multiscale modeling of fluid transport in heterogeneous materials using discrete Boltzmann methods , Mater Struct 2002; 35:650-9

[7] Argentini R, Bakker AF, Lowe CP. Efficiently using memory in lattice Boltzmann Simulations. Future Gener Comput Syst 2004;20(6):973-80

[8] Pan C, Luo LS, Miller CT. An evaluation of lattice Boltzmann schemes for porous medium flow simulation, Comput Fluids 2006;35:898-909

[9] Mattila K, Hyväluoma J, Timonen J, Rossi T. Comparison of implementations of the lattice-Boltzmann method. Comput Math Appl 2008;55(7): 1514-24

[10] Mattila K, Hyväluoma J, Rossi T, Aspnäs M, Westerholm J. An efficient swap algorithm for the lattice Boltzmann method. Comp Phys Comm 2007;176:200-10

[11] Pohl T, Kowarschik M, Wilke J, Iglberger K, Rüde U. Optimization and profiling of the cache performance of parallel lattice Boltzmann codes. Parallel Process Lett 2003;13(4):549-60

[12] Wellein G., Zeiser T, Hager G, Donath S. On the single processor performance of simple lattice Boltzmann kernels. Comput Fluids 2006;35:910-19

[13] Satofuka N, Nishioka T. Parallelization of lattice Boltzmann method for incompressible flow computations. Comput Mech 1999:23:164-71

[14] Kandhai D, Koponen A, Hoekstra AG, Kataja M, Timonen J, Sloot PMA. Lattice-Boltzmann hydrodynamics on parallel systems. Comput Phys Commun 1998;111:14-26

[15] Axner L, Bernsdorf J, Zeiser T, Lammers P, Linxweiler J, Hoekstra AG, Performance evaluation of a parallel sparse lattice Boltzmann solver. J Comp Physics 2008;227:4895-911

[16] Freudiger S, Hegewald J, Krafczyk M. A parallelization concept for a multi-physics lattice Boltzmann prototype based on hierarchical grids. Progr Comput Fluid Dynam Int J 2008;8(1-4):168-78

[17] Wang J, Zhang X, Bengough AG, Crawford JW, Domain-decomposition method for parallel lattice Boltzmann simulation of incompressible flow in porous media. Phys Rev E 2005; 72:016706-11

[18] Bhatnagar PL, Gross EP, Krook M. A model for collision processes in gases. I. Small amplitude processes in charged and neutral one-component systems. Phys Rev 1954;94:511-25

[19] Vidal D. Développement d'algorithmes parallèles pour la simulation d'écoulements de fluides dans les milieux poreux. PhD Thesis, Ecole Polytechnique de Montréal; 2009

[20] mpptest version 1.4b. http://www-unix.mcs.anl.gov/mpi/mpptest/ ;Oct 2008

[21] mpiP version 3.1.2. http://mpip.sourceforge.net/ ;Nov 2008

[22] Auzerais FM, Dunsmuir J, Ferréol BB, Martys N, Olson J, Ramakrishnan TS, Rothman DH, Schwartz LM, Transport in sandstone: a study based on thre dimensional microtomography. Geophys Res Lett 1996;23(7):705-8

[23] Hayes RE, Bertrand F, Tanguy PA. Modelling of fluid/paper interaction in the application nip of a film coater. Transport Porous Media 2000;40:55-72

[24] Vidal D, Ridgway C, Pianet G, Schoelkopf J, Roy R, Bertrand F. Effect of particle size distribution and packing compression on fluid permeability as predicted by lattice-Boltzmann simulations. Comput Chem Eng 2008; doi:10.1016/j.compchemeng.2008.09.003

[25] Pianet G, Bertrand F, Vidal D, Mallet B. Modeling the compression of particle packings using the discrete element method. In proceedings of the 2008 TAPPI Advanced Coating Fundamentals Symposium, Atlanta, GA, USA: TAPPI Press; 2008

4.9 APPENDIX: DEVELOPMENT OF THEORETICAL PARALLEL PERFORMANCE MODELS FOR HETEROGENEOUS POROUS MEDIA

Two cases need to be examined: the speed-up test (constant domain size) and the scale-up test (proportional domain size).

4.9.1 Speed-up test case

The analysis for the speed-up test case is based on the following assumptions:

1) the computer architecture is homogeneous;
2) the processors are linked directly to each other;
3) the sequential fraction of the code is negligible;
4) the computation time comprises all the arithmetic operations performed on fluid nodes (no computations on solid nodes), which are considered equal;
5) the lattice domain size is constant as the number of processors varies and equal to $N_{tot}=n_x \times n_y \times n_z$ (speed-up test).

If the domain of average porosity ε and lattice size N_{tot} is partitioned into n_p subdomains of porosity ε_i and equal lattice size N, we have:

$$\varepsilon = \frac{1}{n_p} \sum_{i=1}^{n_p} \varepsilon_i \tag{4.17}$$

and

$$\varepsilon_i N = \varepsilon_i \frac{N_{tot}}{n_p}, \tag{4.18}$$

where $\varepsilon_i N$ represents the amount of fluid nodes in subdomain i.

If $t_{cal,i}$ denotes the time spent per iteration by processor i to perform m arithmetic (floating-point) computations per fluid node in its associated subdomain, and $t_{com,i}$ the overall time spent per iteration by this processor to transfer $s_{i,j}$ bytes of data to each of its n_i neighboring processors j, we then obtain:

$$t_{cal,i} = \left[\varepsilon_i \frac{N_{tot}}{n_p} m\, t_{oper} \right] \times n_{it} \tag{4.19}$$

and

$$t_{com,i} = \left[2 \sum_{j=1}^{n_i} \left(t_{lat} + s_{i,j}\, t_{data} \right) \right] \times n_{it}, \tag{4.20}$$

where the "2" in Equation (4.20) stands for the number of messages (send/receive) required per processor for each neigboring processor, t_{oper} is the average time spent per arithmetic operation, t_{lat} the communication latency, t_{data} the average time to transfer 1 byte of data, and n_{it} the number of iterations performed to reach the desired solution.

As the CPU time of the parallel LBM code running on n_p processors is limited by the slowest processor, we have:

$$\begin{aligned} t_{CPU,n_p} &= \max_i \left[t_{cal,i} + t_{com,i} \right] \\ &= \max_i \left[\varepsilon_i \frac{N_{tot}}{n_p} m\, t_{oper} + 2 \sum_{j=1}^{n_i} \left(t_{lat} + s_{i,j}\, t_{data} \right) \right] \times n_{it} \end{aligned}, \tag{4.21}$$

If now i=max for the subdomain with the highest porosity, ε_{max}, we then have[40]:

$$t_{CPU,n_p} \approx \left[\varepsilon_{max} \frac{N_{tot}}{n_p} m\, t_{oper} + 2 \sum_{j=1}^{n_{max}} \left(t_{lat} + s_{max,j}\, t_{data} \right) \right] \times n_{it}. \tag{4.22}$$

---

[40] Note that this latter simplification is only true if $\varepsilon_i \frac{N_{tot}}{n_p} m\, t_{oper} >> 2 \sum_{j=1}^{n_i} \left(t_{lat} + s_{i,j}\, t_{data} \right)$.

Also the sequential execution time of the LBM code is given by:

$$t_{CPU,1} = \left[\varepsilon N_{tot} m t_{oper} \right] \times n_{it} . \tag{4.23}$$

The parallel efficiency for a speed-up test is defined as:

$$E(n_p) = \frac{t_{CPU,1}}{n_p \times t_{CPU,n_p}} . \tag{4.24}$$

Combining Equations (4.22) through (4.24) gives the efficiency model for heterogeneous porous media in the case of a speed-up test:

$$E(n_p) = \frac{\varepsilon}{\varepsilon_{max} + \dfrac{2 n_p \sum_{j=1}^{n_{max}} \left(t_{lat} + s_{max,j} t_{data} \right)}{N_{tot} m t_{oper}}} . \tag{4.25}$$

Finally, the parallel computational performance expressed in MLUPS (Millions of Lattice fluid nodes Update Per Second) can be obtained through:

$$P_{comp}(n_p) = \frac{10^{-6} \varepsilon N_{tot} n_{it}}{t_{CPU,n_p}} = \frac{10^{-6} \varepsilon N_{tot}}{\varepsilon_{max} \dfrac{N_{tot}}{n_p} m t_{oper} + 2 \sum_{j=1}^{n_{max}} \left(t_{lat} + s_{max,j} t_{data} \right)} = \frac{10^{-6} n_p}{m t_{oper}} E(n_p) . \tag{4.26}$$

4.9.2 Scale-up test case

The analysis for the scale-up test case is based on the following assumptions:

1) - 4) same as for the speed-up test case;

5) the lattice domain size is proportional to the number of processors: $N_{tot}=n_x \times n_y \times n_z \times n_p$ (scale-up test).

Note that Equations (4.17) through (4.22) still hold, but with the new definition of N_{tot}. However, the sequential execution time of the LBM code with this new definition of N_{tot} now becomes:

$$t_{CPU,1} = \left[\varepsilon \frac{N_{tot}}{n_p} m\, t_{oper} \right] \times n_{it}. \tag{4.27}$$

The parallel efficiency for the scale-up test is now defined as:

$$E\left(n_p\right) = \frac{t_{CPU,1}}{t_{CPU,n_p}}. \tag{4.28}$$

Combining Equations (4.22), (4.27) & (4.28) gives the efficiency model for heterogeneous porous media in the case of a scale-up test:

$$E\left(n_p\right) = \frac{\varepsilon}{\varepsilon_{max} + \dfrac{2 n_p \sum_{j=1}^{n_{max}} \left(t_{lat} + s_{max,j}\, t_{data} \right)}{N_{tot}\, m\, t_{oper}}}. \tag{4.29}$$

As can be seen, Equations (4.25) and (4.29) are identical except for the underlying definition of N_{tot}. Finally, Equation (4.26) holds for the scale-up test with the appropriate definition of N_{tot}.

CHAPITRE 5

ARTICLE 2: A PARALLEL WORKLOAD BALANCED AND MEMORY EFFICIENT LATTICE-BOLTZMANN ALGORITHM WITH SINGLE UNIT BGK RELAXATION TIME

Article soumis en novembre 2008 pour publication dans *Computers & Fluids*.

A Parallel Workload Balanced and Memory Efficient Lattice-Boltzmann Algorithm with Single Unit BGK Relaxation Time

David Vidal[1,2,*], Robert Roy[1] and François Bertrand[1,*]

[1]Ecole Polytechnique de Montréal, Montréal, H3C 3A7, QC, Canada

[2]FPInnovations - Paprican, Pointe-Claire, H9R 3J9, QC, Canada

david.vidal@fpinnovations.ca, {robert.roy, francois.bertrand}@polymtl.ca

5.0 ABSTRACT

A parallel workload balanced and memory efficient lattice-Boltzmann algorithm for Newtonian fluid flow through large porous media is investigated. It relies on a simplified LBM scheme using a single unit BGK relaxation time, which is implemented by means of a shift algorithm and comprises an even fluid node partitioning domain decomposition strategy based on a vector data structure. It provides perfect parallel workload balance, and its two-nearest-neighbour communication pattern combined with a simple data transfer layout results in 20-55% lower communication cost, 25-60% higher computational parallel performance and 40-90% lower memory usage than previously reported LBM algorithms. Performance tests carried out using scale-up and speed-up case studies of Newtonian fluid flow through hexagonal packings of cylinders and a random

* Corresponding authors.

packing of polydisperse spheres on two different computer architectures reveal parallel efficiencies with 128 processors as high as 75% for domain sizes comprising more than 5 billion fluid nodes.

Keywords: Lattice Boltzmann method, fluid flow, porous media, SPMD parallelization, workload balance, memory usage

5.1 INTRODUCTION

The Lattice Boltzmann Method (LBM), developed in the early 90s, is now considered by many researchers as the method of choice for the simulation of single phase or multiphase flows in complex sparse geometries such as porous media [1-3]. It originates from a discretized version of the Boltzmann equation in which collisions are dealt with a relaxation procedure towards local equilibrium. LBM is advantageous with respect to more classical CFD methods because of three main factors: 1) its flexibility in discretizing complex geometries by means of a simple structured lattice into which nodes are marked as "fluid" or "solid" depending on the phase they belong to, 2) the explicit nature of its underlying scheme, which facilitates its parallelization, and 3) its relative ease of implementation. Despite these advantages, three areas of improvement have recently attracted the attention of researchers in relation to the simulation of fluid flow in porous media: 1) the reduction of core memory usage, 2) the improvement of computational efficiency and accuracy of LBM algorithms, and 3) the reduction of workload imbalance often observed with highly heterogeneous domains when computing in parallel.

Several researchers [3-6] showed that transforming the sparse matrix data structure inherent to the LBM lattice into a vector data structure in which only the fluid nodes are retained (no computations take place on the solid nodes) through semi-direct

or indirect addressing can significantly reduce memory usage. For instance, Martys & Hagedorn [3] proposed a simplification of the LBM scheme using a relaxation time equal to unity, which allows a significant reduction of the memory consumption by only requiring the storage of the fluid density and the three components of the velocity for each fluid node in the domain. Despite its memory advantage over standard population-storing methods, this promising density/velocity-storing strategy has not really been picked up by the scientific community, probably because of its apparent restrictions on the relaxation time. Argentini *et al.* [7] introduced another method to reduce memory usage by up to 78%, but it uses a non-BGK approach that is limited to Stokes flows.

Because of LBM young age and heuristic development, a wide range of schemes and implementations are available. Some of them have recently been carefully evaluated by several researchers. For example, Pan *et al.* [8] compared Bhatnagar-Gross-Krook single-relaxation-time (BGK) and multiple-relaxation-time (MRT) LBM schemes, and found a better accuracy with the MRT schemes at the expense of a slightly higher computational cost (10-20%), although the BGK scheme can still provide accurate results when the single-relaxation-time parameter is equal to unity. Very recently, Mattila *et al.* [9] performed a comprehensive comparison in terms of computational efficiency and memory consumption of five different LBM implementations found in the literature: two well-established algorithms, the two-lattice and one-lattice (two-step) algorithms, and three recent ones, the Lagrangian, shift (also called compressed-grid) and swap algorithms. They also investigated the effect of various data structures and memory addressing schemes on computational efficiency. They found that the shift [10] and the newly developed swap algorithms [11], combined with a novel bundle data structure and indirect addressing, both yield high computational performance and low memory consumption. The reader is referred to [9-12] and references therein for a more thorough description of these different implementations.

Concerning parallel efficiency, the classical Cartesian domain decompositions studied by Satofuka & Nishioka [13], which consists in dividing the whole domain in

equally-sized subdomains into slices or boxes, may lead to more and more severe workload imbalance as the number of subdomains increases. As underlined by Pan *et al.* [6], this is due to the variation of the porosity among the subdomains as their number is increased, all the more so in the case of heterogeneous porous media. To alleviate this problem, several methods have been proposed based on orthogonal recursive bisection [6,14], multilevel recursive-bisection or k-way graph partitioning using the METIS package [4,5,15,16]. Although they provide definite improvements with regard to the classical domain decomposition methods, these methods are uneasy to implement, come at high memory expense, create complex communication patterns and can become computationally expensive when dealing with very large systems (say billions of lattice nodes) or when dynamic load balancing is required, such as for fluid flow through packings of highly polydisperse spheres [17] or settling particles [18].

Wang *et al.* [19] proposed a quick, simple and elegant way to balance workload and reduce memory requirement for the two-lattice LBM implementation. The method consists of first vectorizing the data structure through the use of indirect memory addressing as previously proposed by others [4-6]. But to achieve accurate workload balance, the resulting data vector is then simply split into equally-sized subvectors, each of which is assigned to a specific processor. As a result, exact fluid node load balance and high parallel efficiency can be achieved. Furthermore, a simple communication pattern among processors, similar to slice domain decomposition, is obtained since data communication for a given processor only involves its two nearest processors. However, the minimization of the amount of communication is not accounted for by the method, and this may impair its parallel performance. To further improve this method and lower memory usage, we recently proposed a one-lattice algorithm with a vector data structure combined with an even fluid node partitioning domain decomposition and a fully-optimized data transfer layout [20] that carefully selects and minimizes the LBM populations to be communicated. As communication overhead will always impair parallel performance when decreasing domain size, there is an obvious interest in LBM

algorithms that can reduce further the amount of data to be exchanged between processors.

Martys & Hagedorn [3] proposed a few years ago a simplified LBM algorithm that reduces core memory usage. It appears that this method has so far been overlooked by the scientific community, probably because the use of a relaxation time equal to unity is perceived as a severe drawback. In fact, not only does this method decrease core memory usage by only requiring the storage of the density and the three components of the velocity for each fluid node of the domain, but also, something that was not reported by these authors, it reduces by 20-55% (depending on the lattice type) the communication cost as only the fluid density and velocity arrays need to be exchanged instead of the usual 5-9 inward LBM population arrays.

The objective of this work is threefold: 1) to investigate the range of validity of the simplified LBM algorithm proposed by Martys & Hagedorn [3] and assess its applicability, 2) to implement a shift algorithm with a vector data structure and an even fluid node partitioning domain decomposition to improve parallel performance when dealing with heterogeneous domains, and 3) to compare the memory usage and parallel performance of the approach proposed in 2) with theoretical performance model predictions and one-lattice LBM implementations previously studied [20]. First, the lattice Boltzmann method will be recalled and simplifications to this method will be examined. Second, the shift LBM implementation will be described along with the data structure used, and the resulting gain in memory will be evaluated. Parallel communication and workload balance strategies and their resulting performance will be next examined. Finally, the computational advantages of the new proposed method will be assessed by means of two case studies, namely the 3-dimensional fluid flow through hexagonal packings of cylinders and a packing of polydisperse spheres.

5.2 THE LATTICE BOLTZMANN METHOD

Contrary to the conventional CFD methods that solve directly the Navier-Stokes equations, LBM actually "simulates" by means of particles the macroscopic behaviour of fluid molecules in motion. More precisely, LBM comes from the discretization in space (**x**), velocity (**e**) and time (t) of the Boltzmann equation from the kinetic gas theory that describes the evolution of the probability distribution function (or population) of a particle, $f(\mathbf{x},\mathbf{e},t)$, and its microdynamic interactions.

5.2.1 Fused collision-propagation scheme

In practice, the populations of particles propagate and collide at every time step δ_t on a lattice with spacing δ_x and along $\mathbf{e}_i$ velocity directions, where the number of directions i (n_d) depends on the type of lattice. A D3Q15 lattice is used in the present work, i.e a 3-dimensional lattice with n_d=15 velocity directions (Figure 5.1). The general collision-propagation procedure can be mathematically summarized by a fused scheme with a single relaxation time:

$$f_i(\mathbf{x},t) = f_i(\mathbf{x}-\mathbf{e}_i\delta_t, t-\delta_t) - \frac{f_i(\mathbf{x}-\mathbf{e}_i\delta_t, t-\delta_t) - f_i^{eq}(\mathbf{x}-\mathbf{e}_i\delta_t, t-\delta_t)}{\tau^*}, \qquad (5.1)$$

where $f_i(\mathbf{x},t)$ is the particle probability distribution function (or population) in the direction of the velocity $\mathbf{e}_i$ at position **x** and time t, τ^* is a dimensionless relaxation time. The second term of the right-hand side of Equation (5.1) approximates the collision process by means of a single relaxation procedure (the so-called Bhatnager, Gross and Krook approximation [21]) towards a local equilibrium population that can be given for a D3Q15 lattice by

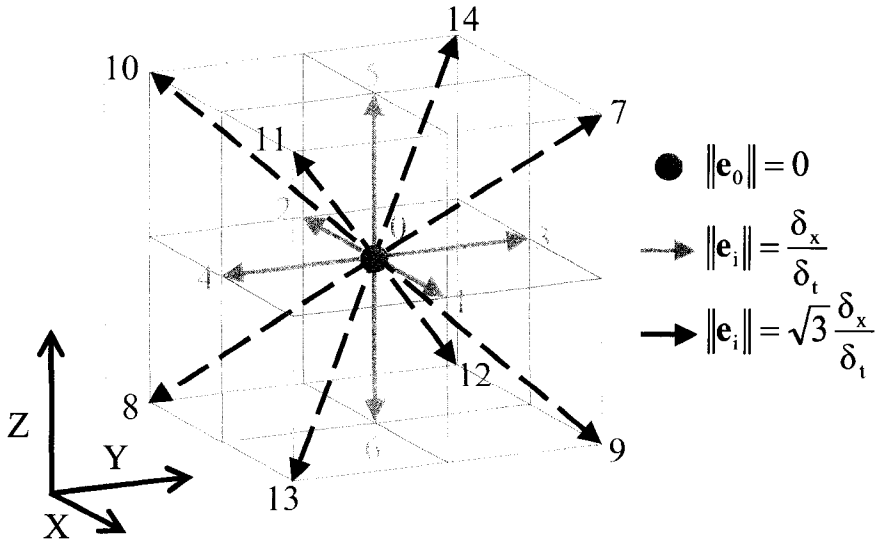


Figure 5.1 – Numbering of the 15 populations of the D3Q15 lattice used in the present work. Odd and even numbers correspond respectively to the forward-pointing and backward-pointing populations. Population 0 is a rest population.

$$f_i^{eq}(\mathbf{x},t) = w_i\rho\left[1+3(\mathbf{e}_i\cdot\mathbf{u})\left(\frac{\delta_t}{\delta_x}\right)^2+\frac{9}{2}(\mathbf{e}_i\cdot\mathbf{u})^2\left(\frac{\delta_t}{\delta_x}\right)^4-\frac{3}{2}(\mathbf{u}\cdot\mathbf{u})\left(\frac{\delta_t}{\delta_x}\right)^2\right], \tag{5.2}$$

where $w_0 = \frac{2}{9}$, $w_i = \frac{1}{9}$ for $i = 1:6$, $w_i = \frac{1}{72}$ for $i = 7:14$, and where the local fluid density is defined as

$$\rho = \rho(\mathbf{x},t) = \sum_i f_i\,, \tag{5.3}$$

and the local macroscopic fluid velocity is given by

$$\mathbf{u} = \mathbf{u}(\mathbf{x},t) = \frac{1}{\rho}\sum_i f_i\mathbf{e}_i\,. \tag{5.4}$$

The dimensionless relaxation time τ^* is related to the kinematic viscosity ν of the fluid:

$$\tau^* = \frac{\nu}{\delta_t c_s^2} + \tfrac{1}{2}\,, \tag{5.5}$$

where c_s is the speed of sound defined as

$$c_s = \sqrt{\frac{3(1-w_0)}{7}}\frac{\delta_x}{\delta_t}. \tag{5.6}$$

It comes from Equations (5.5) & (5.6) that two degrees of freedom among the three parameters τ^*, δ_t and δ_x must be set to obtain a given viscosity. One possibility is to fix δ_x and δ_t, but this can lead to significant inaccuracies, depending on the boundary conditions used, as will be further discussed in Section 5.5.1.

Starting from initial conditions and using appropriate boundary conditions, the collision-propagation scheme described in Equation (5.1) is marched in time until an appropriate convergence is reached (e.g. $(d\mathbf{u}/dt)/(d\mathbf{u}/dt)_{max}=10^{-5}$). The LBM scheme is explicit and the update of the populations at a lattice node is a local operation since it only requires the populations of the immediate neighbouring nodes. This makes the LBM scheme well adapted to distributed parallelization. Finally, as previously mentioned, there are several ways to implement this scheme, which were recently rigorously classified and studied by Mattila *et al.* [9]. In this work, a shift LBM implementation with a vector data structure is used, as described in more detail in Section 5.3.

5.2.2 Simplified fused collision-propagation scheme

It is known (see e.g. [3]) that the choice $\tau^*=1$ leads to a clever and drastic simplification of Equation (5.1) of the LBM scheme. In such a case, the population pointing in the direction of the velocity $\mathbf{e}_i$ at position $\mathbf{x}$ and time step t is only dependent on the population at the local equilibrium at the previous time step at the closest neighbouring node in the $-\mathbf{e}_i$ direction:

$$f_i(\mathbf{x},t) = f_i^{eq}(\mathbf{x}-\mathbf{e}_i\delta_t, t-\delta_t). \tag{5.7}$$

The obvious outcome of this simplified scheme is that the n_d populations at each node no longer need to be stored, and that only the local density and the three components of the fluid velocity must be, as the populations at local equilibrium can directly be computed through Equation (5.2). The standard and simplified schemes will hereafter be called population-storing and density/velocity-storing schemes, respectively. The use of the simplified scheme results in a substantial reduction of memory requirements because of the replacement of n_d arrays by 4 arrays of equal size. On the other hand, this scheme is limited to Newtonian fluid flow problems since non-Newtonian LBM schemes generally entail the determination of local relaxation times that yield the desired local viscosity (e.g. [22]). Moreover, another drawback is that the time step δ_t is now fixed for a given δ_x, so that one cannot play with δ_t to converge faster towards steady state, which may limit the computational performance of the method. However, as will be seen in Section 5.5.1, despite this apparent trade-off between memory and convergence speed, the use of a single unit relaxation time may be fully justified in the case of Newtonian fluids when the accuracy of LBM is considered.

5.2.3 Boundary conditions

In this work, three types of boundary conditions are used, which are typical for a porous medium. First, the boundary conditions at the periphery of the domain are periodic, which means that any out-going population re-enters the domain on its opposite side. Second, to impose a pressure drop ΔP in a given $\mathbf{e}_j$ direction, a body force is added on each node at each iteration in the $\mathbf{e}_i$ directions not normal to $\mathbf{e}_j$. Combined with periodic boundary conditions in this direction, this "trick" enforces the prescribed pressure gradient. Third, the no-slip wall boundary conditions on the solid objects of the

porous domain is modeled using the classical half-way bounce-back method, which reflects any in-coming populations to the wall in the opposite direction at the next iteration. The solid boundary is often reported to be located half-way between the last fluid node and the first solid node, but this is not necessarily true. The accuracy and the choice of these boundary conditions will be further discussed in Section 5.5.1. As opposed to a one-lattice implementation (see e.g. [20]), there is no need here to implement so-called "periodic" and "bounce-back" lattice nodes because velocity and density data at the previous time step can be accessed directly using the shift LBM implementation described next.

5.3 LBM IMPLEMENTATION, DATA STRUCTURE AND MEMORY REQUIREMENTS

Following along the lines of recent other researchers [3-6], a vector data structure is used in this work instead of the sparse matrix data structure inherent to any porous structure, in order to reduce the memory usage by only storing information of the "fluid nodes" since no computations are performed on the "solid nodes" (Figure 5.2). The reader is referred to [20] for a careful comparison of memory usage by these two data structures.

The LBM scheme being explicit, the information at the previous time step is required to compute that at the current time step. To implement this technique and avoid the overwriting of useful information, data at the two previous time steps could be kept in memory, like in the so-called two-lattice algorithm [19], although such an approach is memory inefficient. To alleviate this problem, researchers have devised algorithms requiring only data at the previous time step, which are based on a clever order of the population updates to avoid the overwriting of yet to be propagated populations by ones

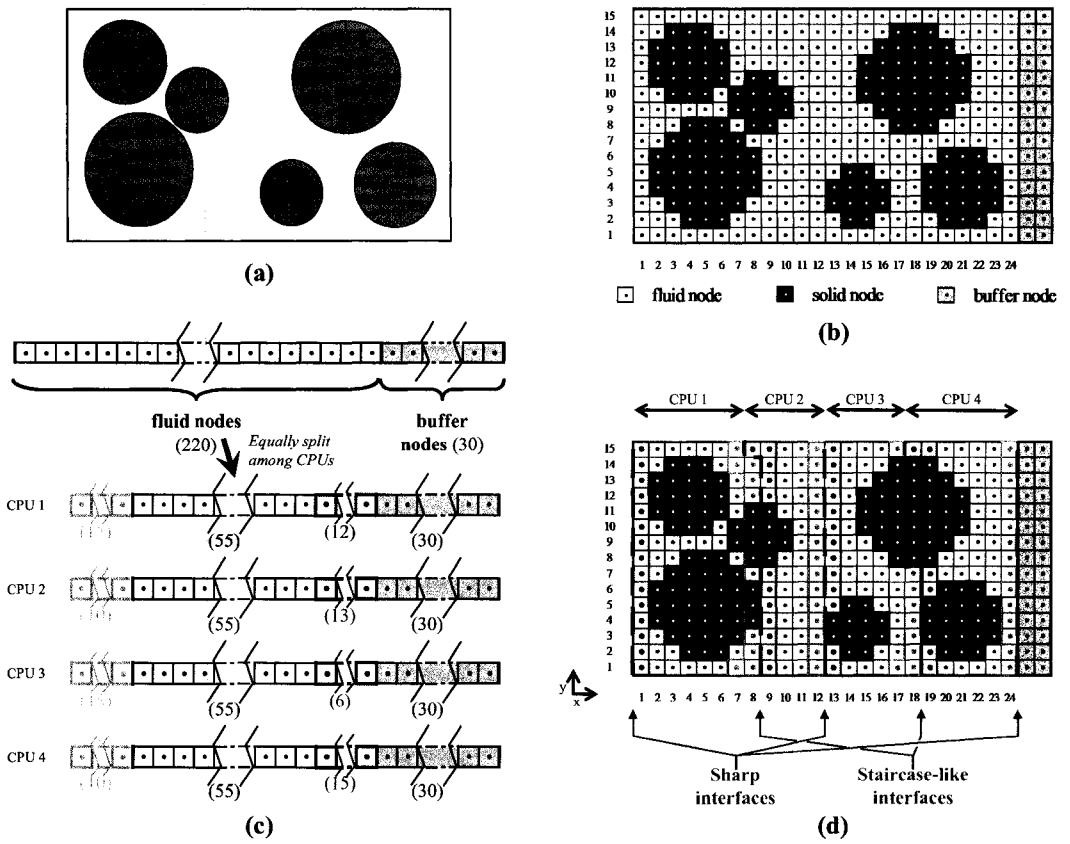


Figure 5.2 – 2-D schematic discretization and decomposition of a porous medium (a) using a 24×15 node domain on a 4-processor cluster. The sparse matrix data structure (b) is converted into a (ordered) vector data structure (c), which is equally split among the processors resulting in a decomposition of the original discretization (d). Note that the vectorization order (y-x order) determines the location of the CPU interfaces (dashed red lines). Green and blue nodes are ghost layer nodes that need to be added to the left and the right of each subdomain, respectively. No computations are performed on ghost layer nodes, which are used to facilitate the transfer of data during the propagation step. Buffer nodes are also added to allow the shift algorithm to proceed without overwriting necessary data (see Figure 5.3).

that have already been. To do so, the one-lattice (two-step) algorithm first executes locally the collisions for all the nodes. It then propagates the resulting populations in ascending node order for the backward-pointing populations and in descending node order for forward-pointing populations. For this to work, it requires the use of so-called (tagged) periodic and bounce-back nodes to store outgoing and solid-pointing populations, respectively [20]. Recently, Mattila *et al.* [11] introduced the "swap" algorithm based on a fused (one-step) propagation-collision scheme that, at each lattice node, permutates populations with neighboring nodes that have not been yet updated

before performing the collision with previously-arrived and newly-permutated populations. Note that, unlike any other algorithms, the swap algorithm does not require the allocation of additional memory, but is restricted to a population-storing scheme. Another possibility is to consider the shift (also called compressed-grid) algorithm, which uses additional memory through buffer nodes to store newly-updated values, thus preventing the overwriting of still useful data [10]. More precisely, the size S_{buf} of the memory offset between specific data at two consecutive time steps is determined by the data spatial dependency, which is related to the propagation step of LBM. As illustrated in Figure 5.2(b) for a 2-D case with periodic boundary conditions, two lattice node columns are necessary to avoid breaking this dependency. In 3-D and similar conditions, a two-layer thick lattice node slice would be required. In practice, the procedure consists of shifting the newly updated data, in reverse (resp. forward) node order, to $+S_{buf}$ (resp. $-S_{buf}$) memory positions at odd (resp. even) time steps, as depicted in Figure 5.3. More details can be found in [10].

As the proposed scheme for a single unit relaxation time (Equation 5.7) consists in a fused scheme, the one-lattice (two-step) algorithm does not represent viable solution and must be ruled out. Moreover, since only density and velocity are to be stored in memory in our scheme, the swap algorithm cannot be used except if additional memory is allocated to store the set of $(\frac{1}{2}n_d - 1)$ populations that are yet to be propagated, the size of which depends on the data spatial dependency, and a trickier update procedure is developed. As a matter of fact, it appears that the shift algorithm, originally developed for a population-storing LBM scheme, can be straightforwardly extended to a density/velocity-storing LBM scheme. For the $(n_x \times n_y \times n_z)$ lattice discretization of a porous domain of porosity ε, it can be shown that the memory requirement (q_{mem}) for the shift LBM implementation with a vector data structure [20] and density/velocity storage is

$$q_{mem} \approx \{8\ \text{bytes}\} \times \underbrace{\left((n_f + S_{buf}) \times 4\right)}_{\text{density/velocity storage}} + \{4\ \text{bytes}\} \times \underbrace{\left(n_f \times (n_d - 1)\right)}_{\text{connectivity list}}$$
$$+ \{2\ \text{bytes}\} \times \underbrace{\left(n_f \times 3\right)}_{\substack{\text{coordinate} \\ \text{list}}} + \{1\ \text{byte}\} \times \underbrace{\left(n_f \times (n_d - 1)\right)}_{\text{bounce-back treatment}}, \tag{5.8}$$

where n_d is the number of populations used in the lattice, which is equal to 15 for a D3Q15 lattice, and $n_f = \left(n_x \times n_y \times n_z\right) \times \varepsilon$ is the number of fluid nodes, {8 bytes} means

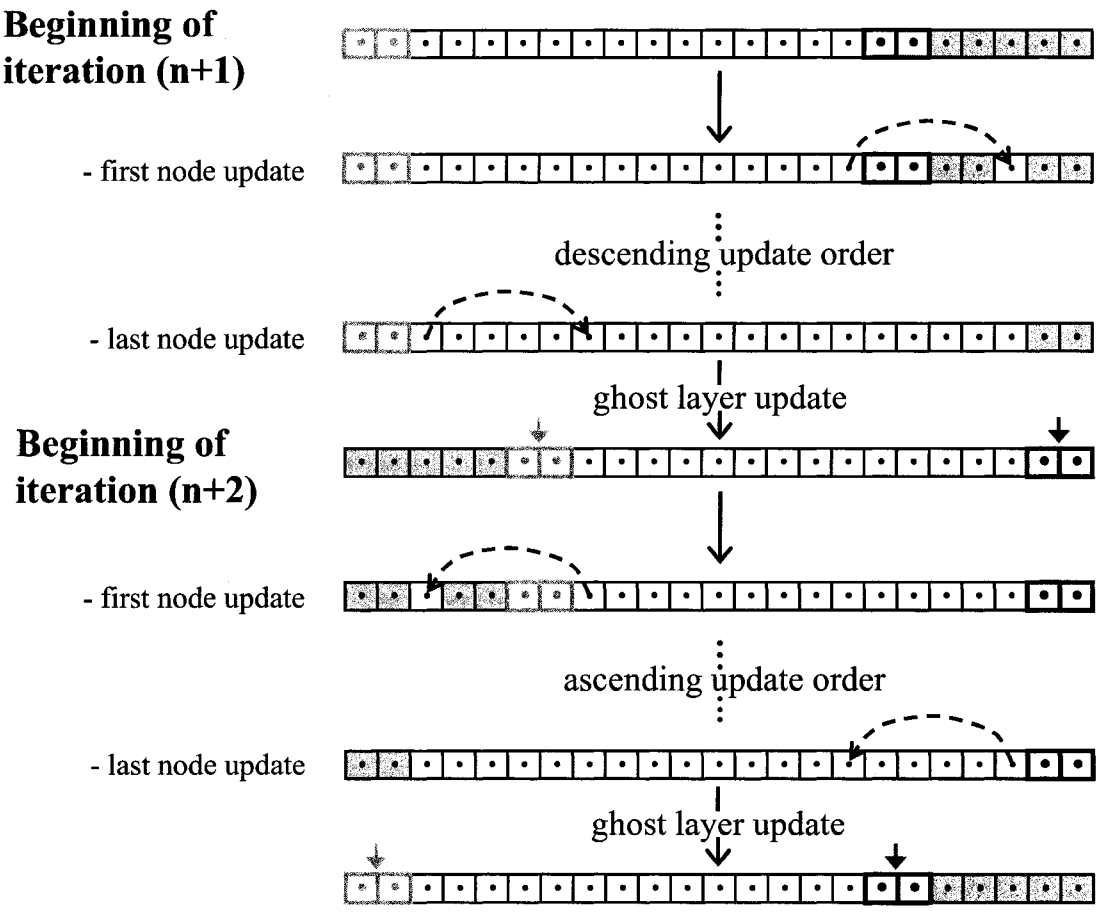


Figure 5.3 – Schematic representation of the shift algorithm procedure at odd (n+1) and even (n+2) time steps for a given subdomain. White and yellow cell nodes represent data (populations or velocity and density) in memory at the beginning of odd and even time steps, respectively. Green and blue nodes are ghost layer nodes that need to be added, to the left and the right of each subdomain, respectively. No computations are performed on ghost layer nodes, which are used to facilitate the transfer of data during the propagation step. Gray cell nodes represent buffer lattice nodes used during shifting to prevent the overwriting of useful data. For the sake of illustration, the spatial data dependency is arbitrarily assumed to be equal to 5, which means that $S_{buf} = 5$.

that the computations are done in double precision and {x byte(s)} refers to an x-byte integer array with $x < 8$. The vector data structure requires indirect addressing through the use of a connectivity list for all fluid nodes. Also, an additional 1-byte array is employed to determine the direction of the incoming populations (the direction is reversed when bounce-back occurs) for the treatment of bounce-back conditions. Finally, it is convenient to store an integer coordinate list of all fluid nodes since such information will be required when post-processing the simulation results.

5.4 PARALLEL WORKLOAD BALANCE & COMMUNICATION STRATEGIES

Three key aspects of a parallel LBM implementation are considered in this section: computational workload, communication overhead and performance.

5.4.1 Workload balance and communication strategies

With a vector data structure, it is rather straightforward to balance workload on a parallel computer, as explained by Wang *et al.* [19]. Each processor receives an equal portion of the vector(s), as illustrated in Figure 5.2(c). Ghost layers for incoming data from neighboring processors as well as buffer nodes to handle the shift procedure are added to each subdomain. Overall, this workload balance procedure leads to a slice domain decomposition method that resembles classical slice decomposition techniques (for instance, y-x vectorization in 2-D and z-y-x vectorization in 3-D create slices in the x direction, as can been seen in Figures 5.2(d) & 5.4, respectively): the data communication pattern is simple due to the fact that each processor needs to

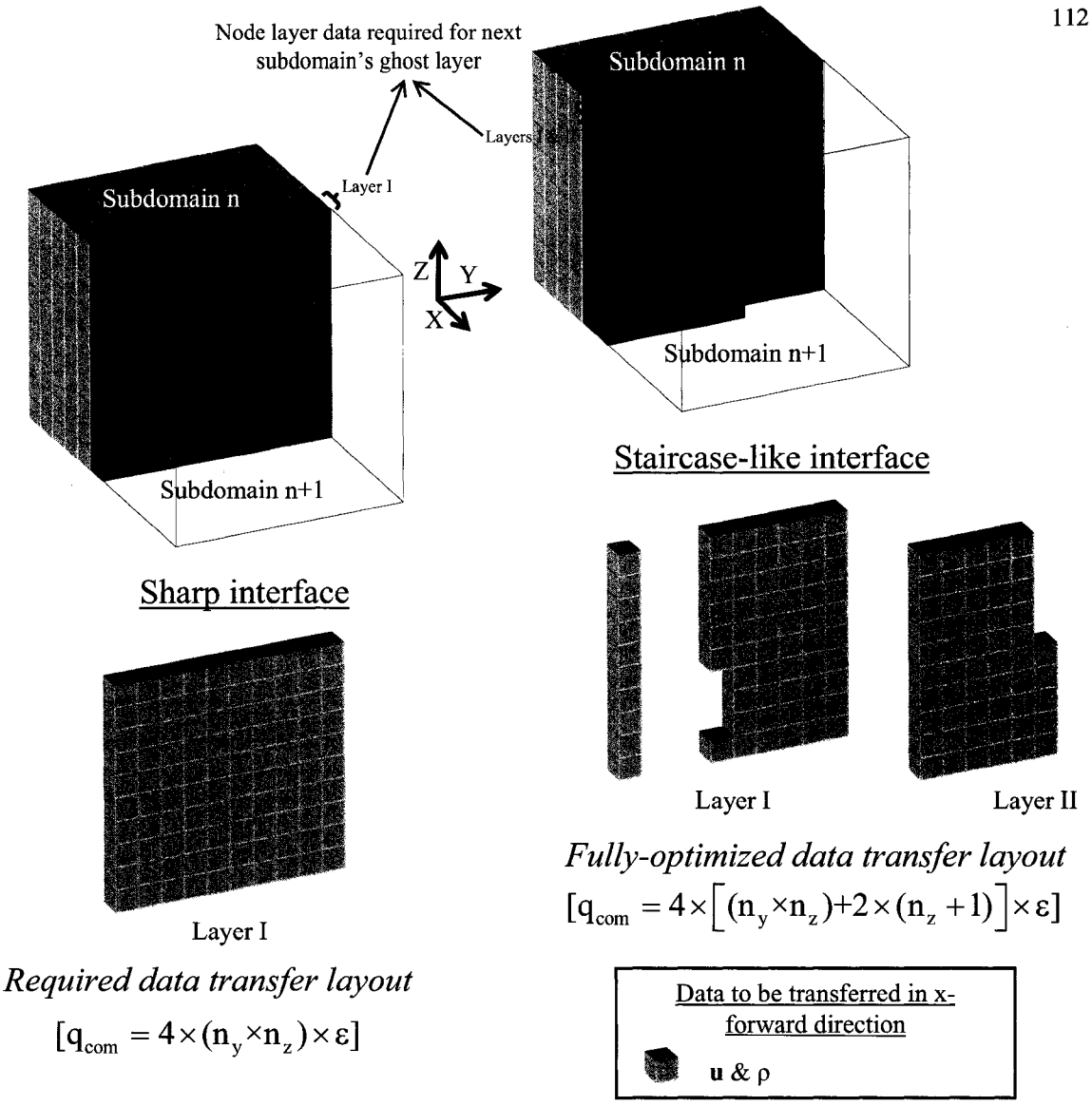


Figure 5.4 – Possible interface scenarios and their corresponding data transfer layouts between subdomain n and subdomain n+1 (transparent) for the density/velocity storing scheme. Note that the vectorization order is z-y-x. Also, these data transfer layouts are valid for periodic boundary conditions and that only data corresponding to fluid nodes need to be exchanged.

communicate with only its two nearest processors. The amount of data to be transferred between processors depends on the position of the interfaces, which can be sharp or staircase-like. In the best-case scenario, the sharp interface, only 4 arrays of size $(n_y \times n_z)\times\varepsilon$ (i.e. the density and the three components of the velocity of layer I in Figure 5.4) are required. In the worst-case scenario, the staircase-like interface, the size of these arrays is larger, $[(n_y \times n_z)+2\times(n_z+1)]\times\varepsilon$, because of the data spatial dependency and the

presence of periodic boundary conditions. The likelihood to be in the worst-case scenario at one interface increases with the number of processors used, and may lead to a communication bottleneck. This data transfer layout is however much simpler than that for a population-storing LBM implementation (see [20]). It requires the transfer of roughly 20-55%[41] fewer data between processors for D3Q15, D3Q19 and D3Q27 lattices. As will be seen, this results in a noticeable improvement of parallel performance when communication overhead is a limiting factor.

5.4.2 Parallel performance

A single-program multiple-data (SPMD) model using MPI and a Fortran compiler was used to implement on parallel computers the proposed shift single unit relaxation time LBM algorithm with a vector data structure combined with even fluid node partitioning. Communications between processors are carried out by non-blocking MPI_ISEND and MPI_IRECV subroutines[42]. When evaluating parallel performance, one can either keep the lattice dimensions constant while increasing the number of processors (a speed-up test) or increase proportionally the lattice dimensions to the number of processors used (a scale-up test). If N_{tot} is the total number of lattice nodes and n_p the number of processors used for a given simulation, these two scenarios lead respectively to $N_{tot}=n_x \times n_y \times n_z$ and $N_{tot}=n_x \times n_y \times n_z \times n_p$. One can then derive for both scenarios the following theoretical efficiency $E(n_p)$ model for a porous media of average porosity ε (see [20] for the development of the model):

[41] ~20% less for D3Q15 and D3Q19 ($4\times[(n_y\times n_z)+2\times(n_z+1)]\times\varepsilon$ vs. $\sim 5\times(n_y\times n_z)\times\varepsilon$) and ~55% less for a D3Q27 ($4\times[(n_y\times n_z)+2\times(n_z+1)]\times\varepsilon$ vs. $\sim 9\times(n_y\times n_z)\times\varepsilon$).

[42] Persistent communications with MPI_SSEND_INIT and MPI_RECV_INIT did not improve performance significantly.

$$E(n_p) = \frac{\varepsilon}{\varepsilon_{max} + \dfrac{q\, n_p (t_{lat} + s\, t_{data})}{N_{tot}\, m\, t_{oper}}} = \frac{\varepsilon}{\varepsilon_{max} + r_{cc}} , \tag{5.9}$$

where ε_{max} is the largest subdomain porosity, m is the number of arithmetic (floating-point) operations per lattice node per iteration, q is the number of MPI_ISEND and MPI_IRECV required per processor (q=4 when a processor has two neighbours), t_{oper} is the average time spent per arithmetic operation, t_{lat} is the communication latency, t_{data} is the average time to transfer 1 byte of data, s is the amount of data that needs to be transferred to one neighbouring processor, equal to $q_{com}\times 8$ bytes, and r_{cc} represents the ratio between the communication and the computational workload. Note that $\varepsilon_{max}=\varepsilon$ when the workload is balanced. The product $(m\, t_{oper})$ for a specific code on a given machine, heretoafter called the nodal computational time, can be approximated by

$$m\, t_{oper} \approx t_{CPU,1} / (n_{it} \times n_f) , \tag{5.10}$$

where $t_{CPU,1}$ is the CPU time measured on a single processor and n_{it} the number of iterations. The latency t_{lat} and data rate transfer t_{data} can be respectively evaluated using utilities such as *mpptest* [23] and *mpiP* [24], although only rough approximations can be obtained because the actual values depend on the number and the size of the messages transferred. From Equation (5.9), it can be seen that the efficiency is bounded by: 1) $\varepsilon/\varepsilon_{max}$ when communication overhead is negligible (i.e. $r_{cc} << \varepsilon_{max}$) and workload is not balanced, 2) 1 when communication overhead is negligible and workload is balanced ($\varepsilon=\varepsilon_{max}$), and 3) $\varepsilon/r_{cc}\rightarrow 0$ when communications become overwhelming (i.e. $r_{cc} >> \varepsilon_{max}$). Furthermore, as shown in [20], the computational performance P_{comp} expressed in MLUPS (Millions of Lattice fluid node Updates Per Second) can be expressed by

$$P_{comp}(n_p) = \frac{10^{-6}\varepsilon\, N_{tot}}{\varepsilon_{max} \dfrac{N_{tot}}{n_p} m\, t_{oper} + q(t_{lat} + s\, t_{data})} = \frac{10^{-6} n_p}{m\, t_{oper}} E(n_p) . \tag{5.11}$$

Interestingly, this equation links the parallel efficiency to the computational performance, which can be evaluated experimentally as

$$P_{comp}(n_p) = \frac{10^{-6} \varepsilon N_{tot} n_{it}}{t_{CPU,n_p}} , \tag{5.12}$$

where t_{CPU,n_p} represents the CPU time with n_p processors. Combining Equations (5.11) & (5.12) gives a way to measure experimentally the parallel efficiency without timings on one single processor:

$$E(n_p) = \frac{\varepsilon N_{tot} n_{it} m t_{oper}}{n_p t_{CPU,n_p}} . \tag{5.13}$$

Note that simulations on one single processor may not be feasible because of the size of the computational domain. In fact, this equation allows to evaluate experimental parallel efficiencies for very large domains. It is valid if the computational time is proportional to the number of lattice nodes, which is rigorously the case for the LBM algorithms considered in this work.

5.5 RESULTS AND DISCUSSIONS

The proposed parallel algorithm relies on a density/velocity-storing single unit relaxation time LBM scheme implemented by means of a shift procedure and an even fluid node partitioning based on a vector data structure. However, the choice of a single unit relaxation time needs to be analyzed with care in conjunction with the boundary conditions used. This will be discussed first in this section. Next, the efficiency of the proposed parallel algorithm will be assessed by means of 3-dimensional fluid flow

simulations through a hexagonal packing of cylinders and a heterogeneous random packing of polydisperse spheres.

5.5.1 Justification of the simplified single-unit-relaxation LBM scheme

It can be inferred from Equations (5.5) & (5.6) that δ_t varies as $O\left(\delta_x^2\left(\tau^* - \frac{1}{2}\right)\right)$ for a prescribed kinematic viscosity ν. Alternatively, one can show that the CPU time scales as $O\left(\left[\delta_x^5\left(\tau^* - \frac{1}{2}\right)\right]^{-1}\right)$. It is thus interesting to use, given δ_x, a large value of τ^* to increase δ_t and converge faster to a desired solution. As will be seen in Section 5.5.2.1, fixing τ^*=1 can save up memory, but at the detriment of increased CPU time. It looks as though

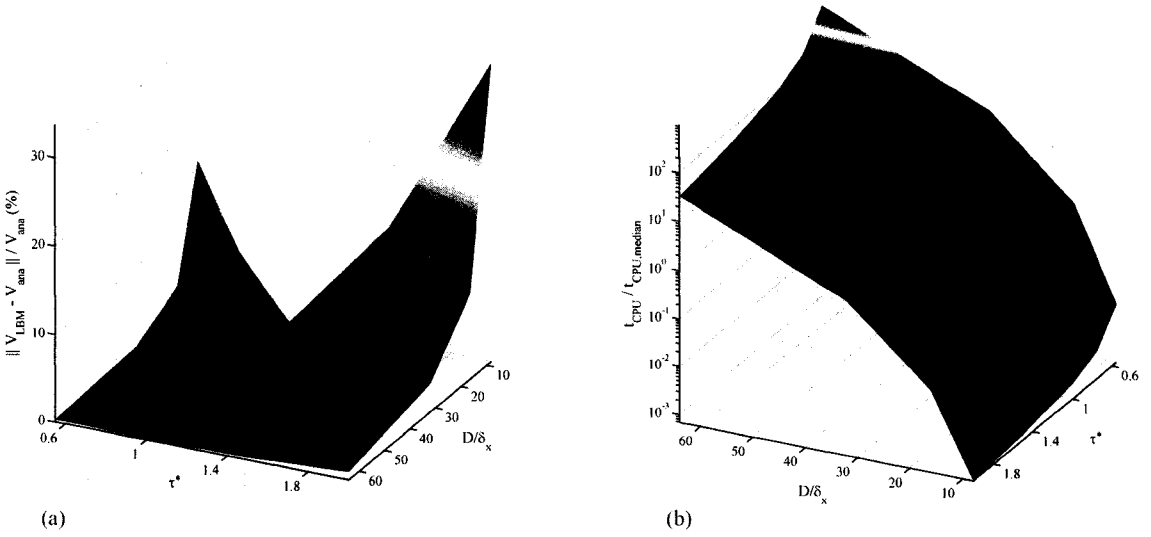


Figure 5.5 – (a) Relative error of the mean velocity with respect to the analytical solution and (b) normalized computation time, as a function of the single dimensionless relaxation time and the ratio of the lateral domain dimension (D) to the lattice size for an LBM fluid flow simulation through a 3-dimensional square duct. The no-slip wall boundary condition is enforced through the half-way bounce-back rule, which gives here a nearly second order accuracy in space ($O(\delta_x^{1.8})$).

the problem consists of finding a trade-off value for τ^*. However, in practice, the choice may be related to the type of boundary conditions.

Many LBM fluid flow problems can be formulated by means of a combination of half-way bounce-back boundary conditions on solid walls and a body force and periodic boundary conditions on the other boundaries. This choice is frequently used in the literature because it is flexible, easy to implement and computationally efficient. Nevertheless, several researchers [25-31] have pointed out the limitations of the half-way bounce-back rule as far as accuracy is concerned. Despite nearly second-order accuracy in space, its overall accuracy depends on τ^*. This is illustrated in Figure 5.5, which presents the relative error of the mean velocity with respect to the analytical solution for a 3-D flow in a square duct using a one-lattice LBM implementation. As shown by others [25-26,29-30], the half-way bounce-back boundary condition guarantees accurate results only for $\tau^*=1$ because the position of the wall depends on τ^* and is precisely half-way when $\tau^*=1$. When τ^* is too large and the lattice resolution is coarse with regard to the geometry, significant errors are made. As a result, if one decides to use the half-way bounce-back boundary condition, one should choose $\tau^*=1$ to obtain accurate results in all circumstances. Note that other methods [25-28,30,32-34] have been developed to strictly enforced no-slip wall boundary conditions independently of τ^*. However, they come at extra computational cost that, although it has never been carefully evaluated, is likely to be proportional to the number of interfaces between the solid and the fluid phases (usually quite high for porous media). Furthermore, some of these methods require extra information to be communicated between processors, which may noticeably impair the parallel performance. This and the fact that half-way bounce-back boundary conditions are rather simple to implement may explain why, despite their limitations, they are still very popular. It also justifies the use of an LBM implementation based on a single unit relaxation time for computing Newtonian fluid flow. Note that one way to alleviate the constraint on CPU time resulting from the choice $\tau^*=1$ and speed up convergence to steady state, is to use the iterative momentum relaxation technique proposed by Kandhai

et al. [35]. This technique was reported to cut down the number of iterations required to reach steady state by 45 to 97%.

5.5.2 Memory and parallel efficiency experiments

The efficiency of the proposed method with regard to parallel performance and memory usage is investigated by means of two case studies that were introduced in a previous article [20]. More precisely, it is compared with two other algorithms described and investigated in that article: 1) a one-lattice LBM implementation with a matrix data structure and a classical slice domain decomposition, and 2) a one-lattice LBM implementation with a vector data structure and an even fluid node partitioning technique combined with a fully-optimized population transfer layout. The latter of the two was found to be the most parallel and memory efficient algorithm in that paper. The reader is referred to [20] for more detailed descriptions of the case studies and these LBM implementations.

5.5.2.1 Hexagonal packings of cylinders

The following numerical experiments were performed on the High-Performance Computing (HPC) cluster (Mammouth(mp)) from the Réseau Québécois de Calcul de Haute Performance (RQCHP). Table 5.1 summarizes the main features of this HPC cluster. The dimension of the lattice was increased proportionally to the number of processors used (scale-up test). More precisely, the total number of lattice nodes (N_{tot}) was $600 \times 10\ n_p \times 347$ in the x, y and z directions, respectively. Four different cylinder radii (R=60.6, 67.1, 73.6 and 80.1 lattice nodes) were investigated in order to vary the

domain porosity. Table 5.2 presents the nodal computational times (m t_{oper}) of the various codes used on Mammouth(mp), as calculated from Equation (5.10). These results clearly show a better computational performance for the shift algorithm. As a matter of fact, Mattila *et al.* [9] also reported a better sequential computational performance for the shift algorithm than for the one-lattice (two-step) algorithm.

Table 5.1 – Specifications of the Mammouth(mp) HPC cluster.

	Mammouth(mp) parallel cluster
Make	Dell PowerEdge SC1425
Processors used	2× 128 Intel Xeon[†]
- clock speed	3.6 GHz
- bus FSB	800 MHz
- RAM	8 GB
- cache	1 MB L2
Interconnection	Infiniband 4x (700-800 MB/s)
- t_{lat} (μs)	~5.0
- t_{data} (ns/byte)	~8.0
Operating system	RedHat Enterprise Linux 4 (2.6.9-42.0.3.ELsmp)
Compiler	Portland Group pgf90 Fortran (6.0-4)
Message passing	MPI (mvapich2 0.9.82)

[†] Only one processor per server was used.

Table 5.2 – Nodal computational time (m t_{oper}) for the various codes on the Mammouth(mp) HPC cluster.

Code	Nodal computational time (ns) with 4-byte integer compiler option	with 8-byte integer compiler option
Proposed shift algorithm with even fluid node vector partitioning	422	N/A
One-lattice algorithm with even fluid node vector partitioning	879	N/A
One-lattice algorithm with classical x-axis slice decomposition	1142	N/A

As displayed in Figure 5.6, the memory requirement model (Equation (5.8)) predicts for the shift algorithm a linear decrease of the memory usage per node as the porosity of the hexagonal packings of cylinders is decreased. The predictions are in good agreement with the numerical data. It can also be observed that the shift algorithm with a single unit relaxation time reduces memory usage by 40 to 75% as compared to the one-lattice LBM implementation with vector data structure. This represents a tour de force considering that this one-lattice implementation is already significantly more memory efficient than the traditional algorithm with a matrix data structure. Note that memory savings could be even greater in the case of D3Q19 or D3Q27 lattice types.

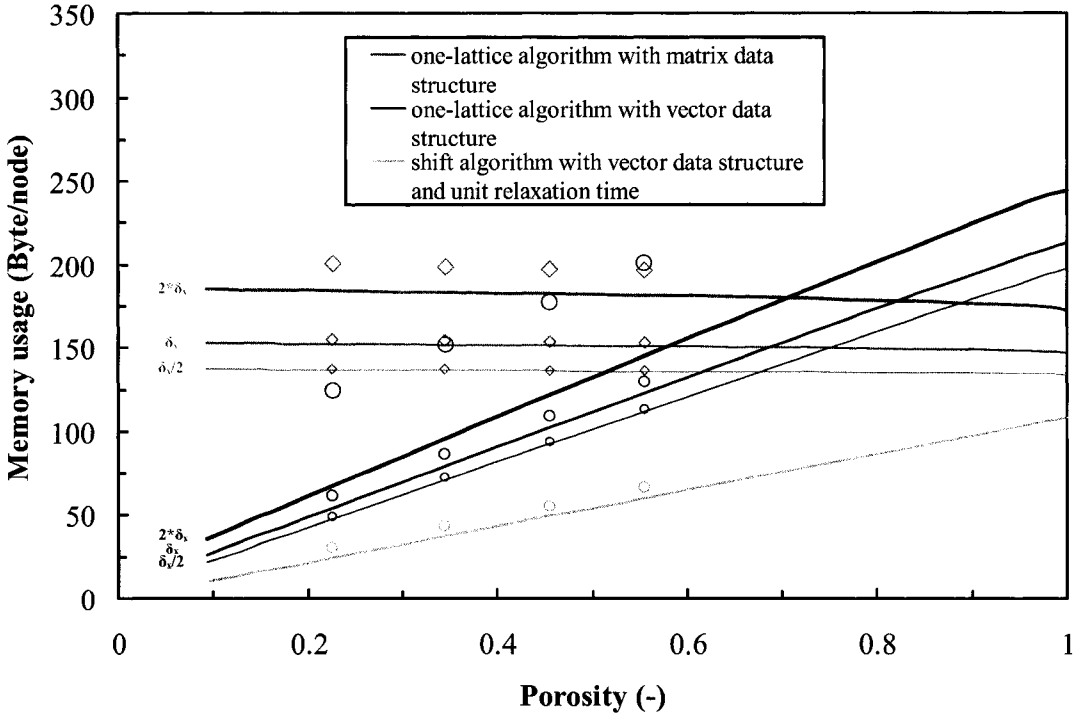


Figure 5.6 – Memory usage per lattice node as a function of the porosity of an hexagonal packing of cylinders for the one-lattice (matrix and vector data structures) and the proposed shift (vector data structure) LBM implementations. The porosity is changed by varying the diameter of the cylinders. Lines correspond to model predictions (Equation (5.8) and models from [20]), and symbols are numerical data points. The thicker the lines or the bigger the symbols, the coarser the lattice (δ_x = 0.0462 μm) for the one-lattice implementations, the memory usage of which depends on the lattice size.

For this scale-up case study, there is an obvious slice domain decomposition along the cylinder axis (y axis), which can lead to high parallel efficiency due to straightforward workload balance and constant communication load (involving 600×347 lattice nodes). However, to emphasize the improvement obtained by resorting to an even fluid node vector partitioning domain decomposition when load imbalance is present, the computational domain was discretized and decomposed in the x direction, which is perpendicular to the cylinder axial direction. Figure 5.7 presents the parallel efficiency and computational performance for the proposed method and the two above-mentioned one-lattice LBM implementations. As expected, the algorithms using even fluid node vector partitioning significantly improve the parallel efficiency with respect to the classical domain decompositions. In particular, the drop and fluctuations in efficiency resulting from variations in subdomain porosity for classical domain decomposition are suppressed. This proves the workload balancing capability of the even fluid node vector partitioning method. Moreover, the efficiency model (Equation (5.9)) predicts well the different trends observed experimentally. Surprisingly, the efficiency of the proposed shift algorithm (*blue curve with diamonds*) is slightly lower than that of the one-lattice counterpart (*red curve with circles*). In fact, the ~20% lower communication load (see Section 5.4.1) is outshined by the better intrinsic computational performance of the shift algorithm. This result is due to a higher r_{cc} ratio for the shift algorithm (see Equation (5.9)). Nevertheless and more importantly, this proposed algorithm provides in this scale-up case study a sustained 25% increase in computational parallel performance, as evidenced in Figure 5.7, at a much lower memory usage.

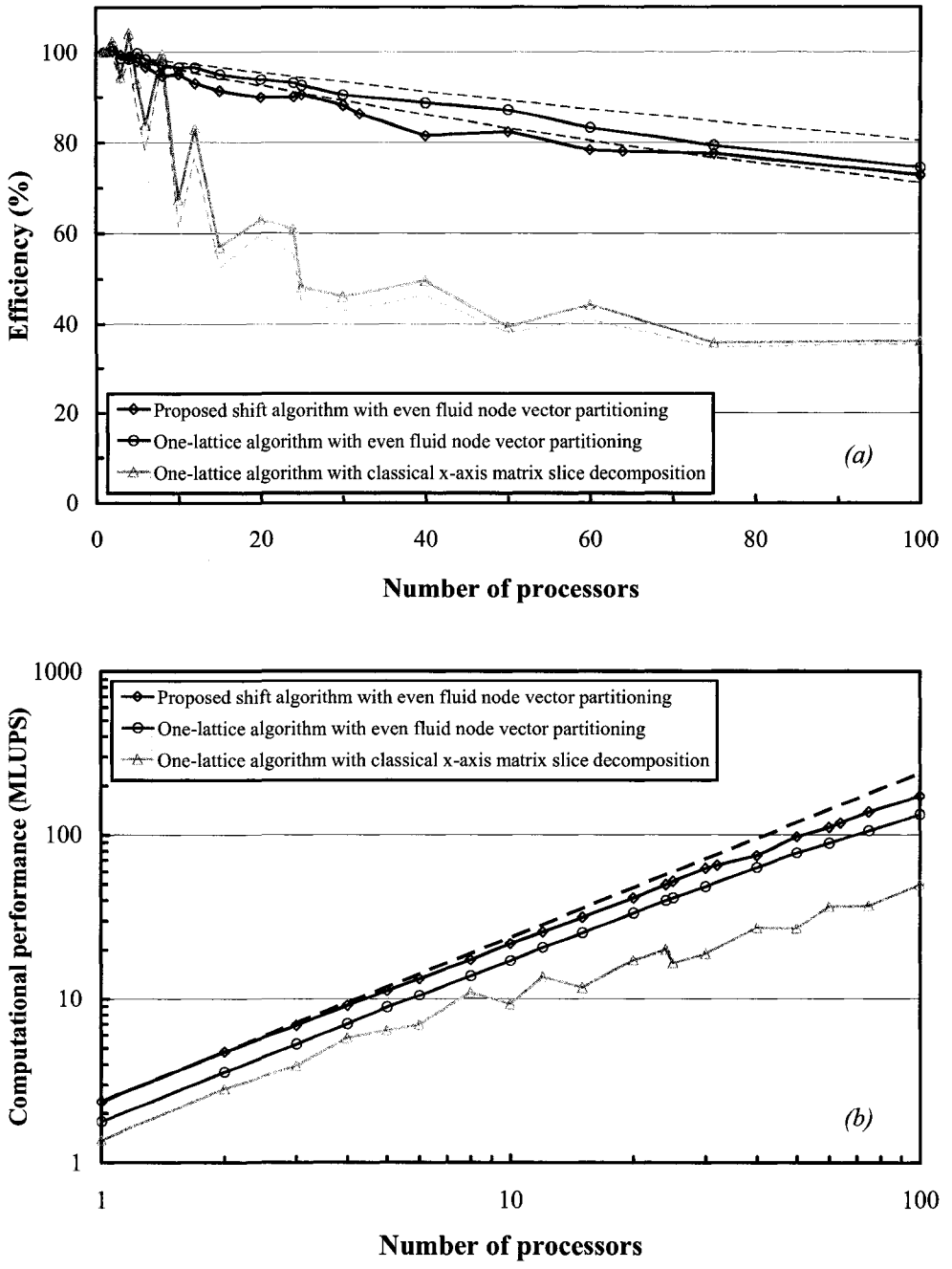


Figure 5.7 – Parallel efficiency *(a)* and computational performance *(b)* comparisons on the Mammouth(mp) cluster between the one-lattice and shift algorithms with x-slice and even fluid node vector partitioning domain decompositions for an hexagonal packing of cylinders (R=73.6 lattice nodes) with proportional domain size (scale-up test). The *colored dashed lines* in *(a)* represent the model predictions from Equation (5.9) for the corresponding algorithms. In *(b)*, the *black dashed line* represents the theoretical linear performance for the shift algorithm with an even fluid node vector partitioning domain decomposition.

5.5.2.2 Random packing of polydisperse spheres

To further investigate the parallel performance and the memory gain of the proposed method, a speed-up test was carried out on a 3-dimensional random packing of polydisperse spheres (ε=27.5%) generated using a Monte-Carlo packing procedure described elsewhere [17]. The domain size (400^3 lattices nodes) can fit in the memory of a single server and was kept constant as the number of processors was increased (speed-up test). These tests were performed on the HPC cluster (Artemis) from FPInnovations. Table 5.3 summarizes the main features of this HPC cluster and Table 5.4 displays the nodal computational times of the various codes, which evidences here again the intrinsic computational superiority of the shift algorithm.

As can be seen from Table 5.5, the combination of a vector data structure and a single unit relaxation scheme reduces by a factor of 5.3 with respect to a one-lattice algorithm with classical slice decomposition the memory required to simulate the flow through the packing. The single unit relaxation scheme alone reduces the memory by a factor of 3.1, which is fairly close to the factor of 3.75 theoretically achievable by replacing the storage of 15 populations by that of the density and 3 velocity components.

Figure 5.8 shows a significant decrease in parallel performance as the number of processors increases for the three algorithms investigated. This is explained by the reduction of the subdomain granularity as the number of processors increases, which eventually leads to an overwhelming communication overhead. In particular, it can be seen that the proposed shift implementation (*blue curve with diamonds*) underperforms as compared to its one-lattice counterpart (*red curve with circles*). As already explained in Section 5.5.2.1, this is due to the higher communication/computational workload ratio (r_{cc}) resulting from the better intrinsic computational performance of the shift algorithm. These trends are confirmed by the efficiency model predictions, although the quantitative

Table 5.3 – Specifications of the Artemis HPC cluster.

	Artemis parallel cluster
Make	Dell PowerEdge 1950
Processors used	2× 64 quad core Intel Xeon 5440
- clock speed	2.83 GHz
- bus FSB	1333 MHz
- RAM	16 GB
- cache	12 MB L2
Interconnection	Gigabit Ethernet
- t_{lat} (μs)	~1.0
- t_{data} (ns/byte)	~12.0
Operating system	CentOS 4.6 (2.6.9-67.0.15.ELsmp)
Compiler	Intel Fortran (10.1.015)
Message passing	MPI (OpenMPI 1.2.6)

Table 5.4 – Nodal computational time (m t_{oper}) for the various codes on the Artemis HPC cluster.

Code	Nodal computational time (ns) with 4-byte integer compiler option	with 8-byte integer compiler option
Proposed shift algorithm with even fluid node vector partitioning	197	250
One-lattice algorithm with even fluid node vector partitioning	443	478
One-lattice algorithm with classical x-axis slice decomposition	704	802

Table 5.5 – Memory usage of the various codes on a single processor for the spherical particle packing case study.

Code	Memory usage (GB)
Proposed shift algorithm with even fluid node vector partitioning	1.9
One-lattice algorithm with even fluid node vector partitioning	5.9
One-lattice algorithm with classical x-axis slice decomposition	10.1

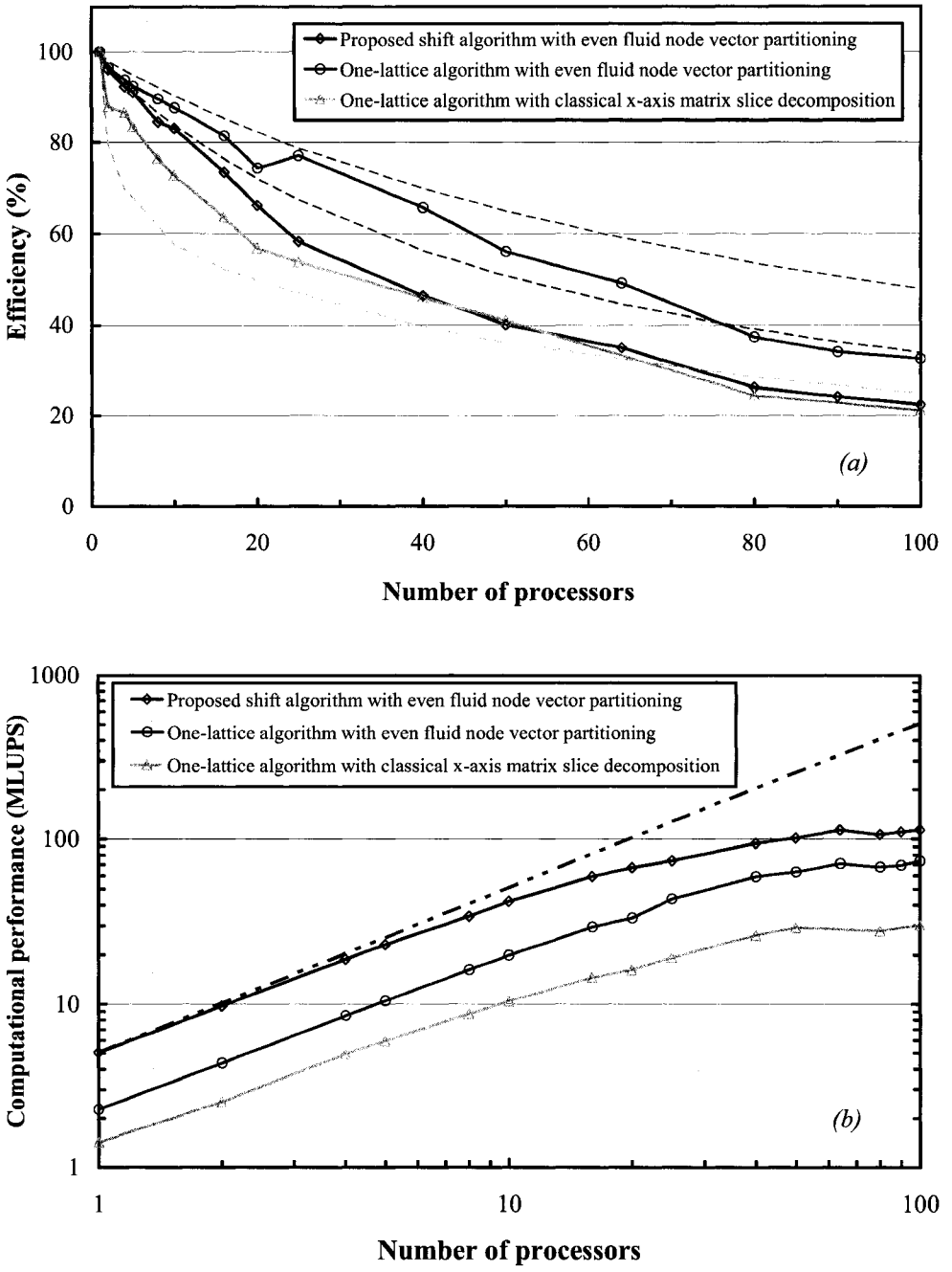


Figure 5.8 – Parallel efficiency *(a)* and computational performance *(b)* comparisons on the Artemis cluster between the one-lattice and shift algorithms with x-slice Cartesian and even fluid node vector partitioning domain decompositions for a random packing of polydisperse spheres with constant domain size (speed-up test). The *colored dashed lines* in *(a)* represent the model predictions from Equation (5.9) for the corresponding algorithms. In *(b)*, the *black dashed line* represents the theoretical linear performance for the shift algorithm with an even fluid node vector partitioning domain decomposition.

agreement with the numerical data is not as good as in the previous case study. This can be attributed to the small size of the domain investigated.

To assess the performance improvement and the adequacy of the efficiency models when the domain size is increased, a scale-up test from 384^3 to 2630^3 node lattices corresponding to a maximum of 5.0×10^9 fluid nodes was carried out. The computational performance and parallel efficiency for 128 processors was calculated through Equations (5.12) & (5.13) for the three algorithms (Figure 5.9). Note that for a domain size larger than 1280^3 lattice nodes, the 8-byte integer compilation option was needed for the indexing of nodes at the pre-processing stage. As expected, the results show an increasing performance and efficiency as the domain size and the corresponding granularity are increased. Moreover, the model predictions from Equations (5.9) & (5.11) (*color dotted lines*) become extremely good above $\sim1.2\times10^8$ fluid nodes (768^3 node lattices). Here again, the parallel efficiency of the one-lattice algorithm with even fluid node vector partitioning surpasses the proposed shift algorithm at the same domain size, but the difference decreases as the domain size increases. As both algorithms follow closely the performance models, it can be concluded that they exhibit the expected workload balance that tends towards 100% ($E\rightarrow1$) as the domain size is increased. In fact, efficiencies as high as 79% and 75% for 128 processors are obtained for the largest domain with the one-lattice and shift algorithms, respectively. Despite its slightly lower efficiency, it can be observed in Figure 5.9 that the computational parallel performance of the shift algorithm is about 60% superior, and that the slope of the curve is significantly steeper than that of the one-lattice algorithm. This difference in performance between the two algorithms is significantly larger than in the case of the (older) Mammouth(mp) architecture (see Figure 5.8) because of a larger cache and faster bus (see Tables 5.1 & 5.3). Unsurprisingly, the use of 8-byte integers affects the computational performance, all the more so for the shift algorithm due to a higher proportion of integer-based operations. Moreover, the efficiency of the one-lattice algorithm with a classical x-axis slice decomposition, which is significantly lower than that for the other two methods, levels off as domain size is increased to a lower

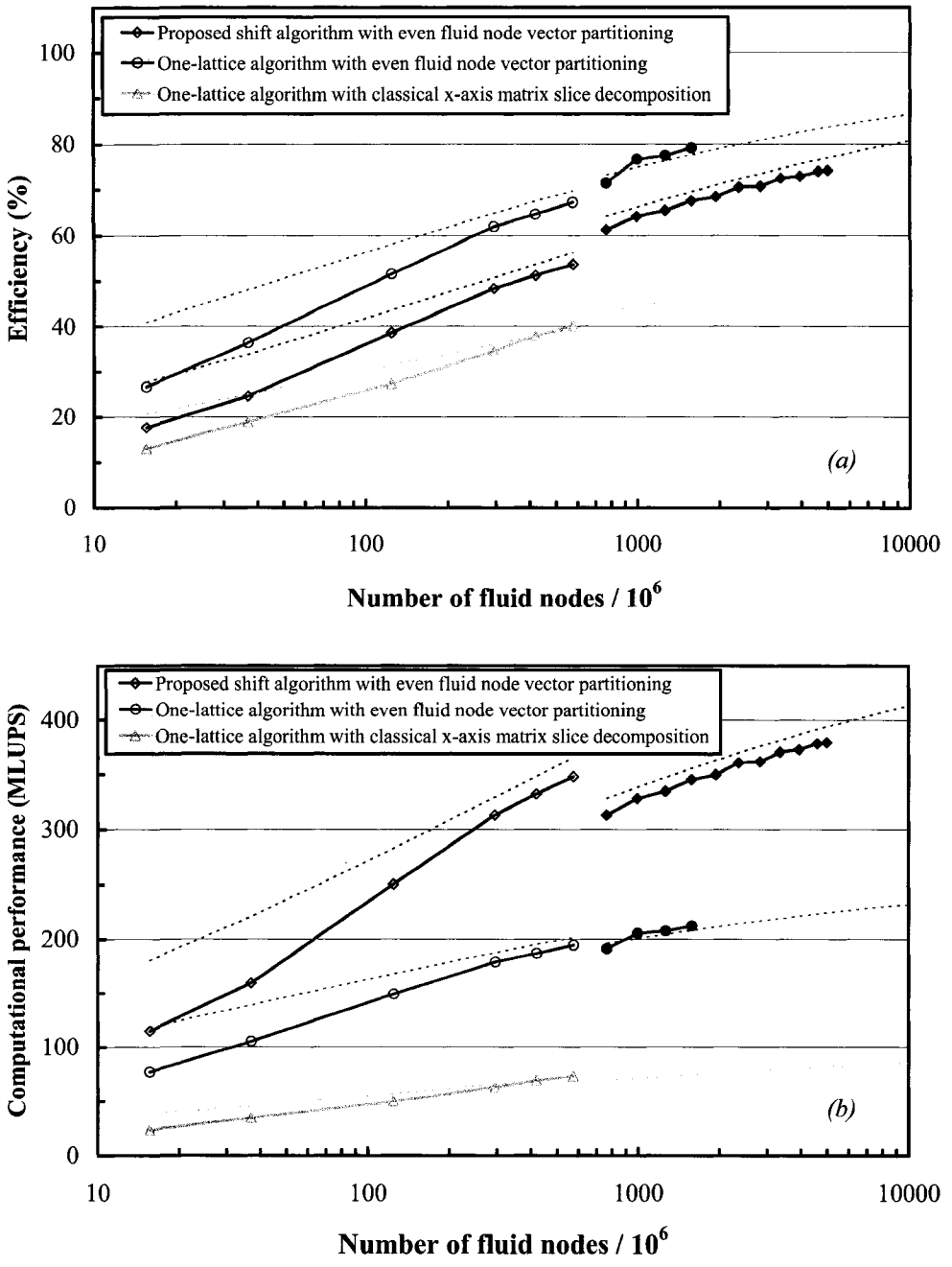


Figure 5.9 – Parallel efficiency (*a*) and computational performance *(b)* comparisons at 128 processors on the Artemis cluster between the one-lattice and shift algorithms with x-slice and even fluid node vector partitioning domain decompositions for a random packing of polydisperse spheres with increasing domain size (scale-up test). The *colored dotted lines* represent the model predictions from Equations (5.13) & (5.12) for the corresponding domain decompositions. Open and filled symbols correspond to simulations performed respectively with 4- and 8-byte integers.

asymptotical value equal to $\varepsilon/\varepsilon_{max}$=27.5%/42.2%=65.2%. Quite clearly, this compromises the applicability of the method for large domain sizes. Finally, on HPC cluster Artemis and its 1024 GB memory, the largest domains tested and that could fit in memory comprised 5.0×10^9, 1.6×10^9 and 5.8×10^8 fluid nodes, for the shift, the one-lattice with even fluid node vector partitioning and the one-lattice with classical slice decomposition methods, respectively. In other words, these two one-lattice implementations are surpassed by the shift algorithm by factors of 3.1 and 10, respectively.

5.6 CONCLUDING REMARKS

An efficient parallel LBM algorithm was introduced for simulating Newtonian fluid flow through porous media. It provides perfect parallel workload balance owing to a two-nearest-neighbour communication pattern and a simple lattice-type-independent data transfer layout with lower (20-55%) communication cost and higher (25 to 60%, depending on the architecture used) computational parallel performance than previously reported LBM algorithms. With this algorithm, the usual trade-off between memory and computational performance is overstepped owing to a 40-90% reduction in memory usage with respect to classical population-storing LBM algorithms. The proposed algorithm is built around four combined strategies to achieve remarkable performance. First, a vector data structure is used instead of the sparse matrix structure inherent to porous media in order to reduce memory requirements. Second, taking advantage of this vector data structure, an even fluid node partitioning domain decomposition technique is used to perfectly balance the parallel workload. Third, the use of a single unit relaxation time simplifies the collision-propagation LBM scheme by replacing the usual population array storage by a density/velocity array storage, which leads to significant memory savings and reduces parallel communication cost. Finally, to further reduce the memory

usage and improve the sequential computational performance, a shift algorithm that overlaps the data related to two consecutive time steps into smaller memory space while accounting for their spatial dependency was implemented. If resorting to a single unit relaxation time may appear restrictive at first sight, its use is fully justified as far as accuracy is concerned when classical, computationally efficient, half-way bounce-back boundary conditions are considered. Indeed, it is shown in this work that accurate results are only guaranteed for such boundary conditions when a relaxation time equal to unity is employed. The major drawback of the proposed algorithm is that it is restricted to Newtonian fluid flows because non-Newtonian fluid LBM schemes are generally based on variable local relaxation times to achieve variable local viscosities. For such flows, one-lattice, shift or swap algorithms with variable relaxation times and even fluid node vector partitioning domain decomposition remain the methods of choice, although much less memory efficient than the method proposed in this work.

The memory and parallel computational performances were assessed on two different computer architectures by means of scale-up and speed-up case studies for Newtonian fluid flows through hexagonal packings of cylinders and a random packing of polydisperse spheres. Numerical data were observed to be in very good agreement with performance model predictions, showing that the algorithm parallel efficiency tends asymptotically to 100% as domain size is increased, and that workload is thus well-balanced despite the heterogeneity of the domains tested. Efficiencies with 128 processors as high as 75% were found for domain sizes comprising as many as 5 billion fluid nodes. To our knowledge, this is the first time that such large flow simulations, which required overall less than 1 TB of memory for the largest domain sizes, are reported. This highlights the memory efficiency of the proposed algorithm. The domain sizes investigated also justify the use of the even fluid node partitioning domain decomposition instead of more advanced techniques such as spectral recursive bisection or k-way graph partitioning, as these would be unpractical for such large systems.

5.7 ACKNOWLEDGMENTS

The computer resources and support from the Réseau Québécois de Calcul de Haute Performance (RQCHP) and from FPInnovations as well as the financial contribution of the NSERC Sentinel Network are gratefully acknowledged. Special thanks to Louis-Alexandre Leclaire and Christian Poirier.

5.8 REFERENCES

[1] Succi S. The lattice Boltzmann Equation for Fluid Dynamics and Beyond. Oxford, UK: Oxford Science Publications; 2001

[2] Nourgaliev RR, Dinh TN, Theofanous TG, Joseph D. The lattice Boltzmann equation method: theoretical interpretation, numerics and implications. Int J Multiphase Flow 2003;29(1):117-69

[3] Martys NS, Hagedorn JG. Multiscale modeling of fluid transport in heterogeneous materials using discrete Boltzmann methods , Mater Struct 2002; 35:650-9

[4] Dupuis A, Chopard B. An object oriented approach to lattice gas modeling. Future Gener Comput Syst 2000;16(5):523-32

[5] Schulz M, Krafczyk M, Tolke J, Rank E. Parallelization strategies and efficiency of CFD computations in complex geometries using lattice Boltzmann methods on high performance computers. In: Breuer M, Durst F, Zenger C, editors. High performance scientific and engineering computing. Berlin: Springer Verlag; 2002. p.115-22

[6] Pan C, Prins JF, Miller CT. A high-performance lattice Boltzmann implementation to model flow in porous media. Comput Phys Commun 2004; 158:89-105

[7] Argentini R, Bakker AF, Lowe CP. Efficiently using memory in lattice Boltzmann Simulations. Future Gener Comput Syst 2004;20(6):973-80

[8] Pan C, Luo LS, Miller CT. An evaluation of lattice Boltzmann schemes for porous medium flow simulation, Comput Fluids 2006;35:898-909

[9] Mattila K, Hyväluoma J, Timonen J, Rossi T. Comparison of implementations of the lattice-Boltzmann method. Comput Math Appl 2008;55(7): 1514-24

[10] Pohl T, Kowarschik M, Wilke J, Iglberger K, Rüde U. Optimization and profiling of the cache performance of parallel lattice Boltzmann codes. Parallel Process Lett 2003;13(4):549-60

[11] Mattila K, Hyväluoma J, Rossi T, Aspnäs M, Westerholm J. An efficient swap algorithm for the lattice Boltzmann method. Comp Phys Comm 2007;176:200-10

[12] Wellein G., Zeiser T, Hager G, Donath S. On the single processor performance of simple lattice Boltzmann kernels. Comput Fluids 2006;35:910-19

[13] Satofuka N, Nishioka T. Parallelization of lattice Boltzmann method for incompressible flow computations. Comput Mech 1999:23:164-71

[14] Kandhai D, Koponen A, Hoekstra AG, Kataja M, Timonen J, Sloot PMA. Lattice-Boltzmann hydrodynamics on parallel systems. Comut Phys Commun 1998;111:14-26

[15] Axner L, Bernsdorf J, Zeiser T, Lammers P, Linxweiler J, Hoekstra AG, Performance evaluation of a parallel sparse lattice Boltzmann solver. J Comp Physics 2008;227:4895-911

[16] Freudiger S, Hegewald J, Krafczyk M. A parallelization concept for a multi-physics lattice Boltzmann prototype based on hierarchical grids. Progr Comput Fluid Dynam Int J 2008;8(1-4):168-78

[17] Vidal D, Ridgway C, Pianet G, Schoelkopf J, Roy R, Bertrand F. Effect of particle size distribution and packing compression on fluid permeability as predicted by lattice-Boltzmann simulations. Comput Chem Eng 2008; doi:10.1016/j.compchemeng.2008.09.003

[18] Pianet G, Bertrand F, Vidal D, Mallet B. Modeling the compression of particle packings using the discrete element method. In proceedings of the 2008 TAPPI Advanced Coating Fundamentals Symposium, Atlanta, GA, USA: TAPPI Press; 2008

[19] Wang J, Zhang X, Bengough AG, Crawford JW, Domain-decomposition method for parallel lattice Boltzmann simulation of incompressible flow in porous media. Phys Rev E 2005; 72:016706-11

[20] Vidal D, Roy R, Bertrand F. On improving the performance of large parallel lattice Boltzmann flow simulations in heterogeneous porous media. submitted to Comput Fluids

[21] Bhatnagar PL, Gross EP, Krook M. A model for collision processes in gases. I. Small amplitude processes in charged and neutral one-component systems. Phys Rev 1954;94:511-25

[22] Gabbanelli SG, Drazer G, Koplik J. Lattice Boltzmann method for non-Newtonian fluid flows. Phys Rev E 2006;72:046312

[23] mpptest vesion 1.4b. http://www-unix.mcs.anl.gov/mpi/mpptest/ ;Oct 2008

[24] mpiP version 3.1.2. http://mpip.sourceforge.net/ ;Nov 2008

[25] Noble DR, Chen S, Georgiadis JG, Buckius RO. A consistent hydrodynamic boundary condition for the lattice Boltzmann method. Phys Fluids 1995; 7(1): 203-9

[26] Inamuro T, Yoshino M, Ogino F. A non-slip boundary condition for the lattice Boltzmann simulations. Phys Fluids 1995;7(12): 2928–30 & Erratum: Phys Fluids 1996;8(4):1124

[27] Maier R, Bernard RS, Grunau DW. Boundary conditions for the lattice Boltzmann method. Phys Fluids, 1996;8(7):1788-1801

[28] Chen S, Martinez D, Mei R. On boundary conditions in lattice Boltzmann methods. Phys Fluids 1996; 8(9):2527-36

[29] Gallivan MA, Noble DR, Georgiadis JG, Buckius RO. An evaluation of the bounce-back boundary condition for lattice Boltzmann simulations. Int J Numer Meth Fluids 1997;25:249-63

[30] Zou Q, He X. On pressure and velocity boundary conditions for the lattice Boltzmann BGK model. Phys Fluids 1997;9(6):1591-8

[31] Holdych DJ, Noble D, Georgiadis J, Buckius R. Truncation error analysis of lattice Boltzmann methods. J Comput Phys 2004;193:595–619

[32] Chopard B, Dupuis A. A mass conserving boundary condition for lattice Boltzmann models. Int J Mod Phys B 2003;17:103-7

[33] Guo A, Zheng C, Shi B. An extrapolated method for boundary conditions in lattice Boltzmann method. Phys Fluids 2002;14(6):2007-10

[34] Fang HP, Chen SY. Lattice Boltzmann method for three-dimensional moving particles in a Newtonian fluid. Chinese Phys 2004;13(1):47-53

[35] Kandhai D, Koponen A, Hoekstra A, Sloot PMA. Iterative momentum relaxation for fast lattice-Boltzmann simulations. Future Generation Comp Syst 2001;18:89-96

CHAPITRE 6

ARTICLE 3: EFFECT OF PARTICLE SIZE DISTRIBUTION AND PACKING COMPRESSION ON FLUID PERMEABILITY AS PREDICTED BY LATTICE-BOLTZMANN SIMULATIONS

Publié dans *Computers & Chemical Engineering* 33(2009) 256-266

S'est vu décerné le prix de la meilleure communication scientifique au *2008 TAPPI Advanced Coating Fundamentals Symposium*

Effect of Particle Size Distribution and Packing Compression on Fluid Permeability as Predicted by Lattice-Boltzmann Simulations[†]

David Vidal[1,2,‡], Cathy Ridgway[3], Grégoire Pianet[1], Joachim Schoelkopf[3], Robert Roy[1] and François Bertrand[1,‡]

[1†]Ecole Polytechnique de Montréal, Montréal, H3C 3A7, QC, Canada

[2]FPInnovations - Paprican, Pointe-Claire, H9R 3J9, QC, Canada

[3]OMYA AG, CH-4665 Oftringen, Switzerland

6.0 ABSTRACT

Massive parallel lattice Boltzmann method simulations of flow through highly polydispersed spherical particle packings formed using Monte-Carlo methods were performed. The computed fluid permeabilities were compared to experimental data obtained from blocks made of three natural ground calcium carbonate powders compressed at different levels. The agreement with experimental measurements is excellent considering the approximations made. A series of flow simulations was also performed for packings of spherical particles compressed at different levels with increasing polydispersity modeled with both lognormal and Weibull size distributions. The predicted permeabilities were found to follow reasonably well the Carman-Kozeny correlation although an increasing deviation towards lower predicted permeabilities with

---

[†] Based on a presentation at the 2008 TAPPI Advanced Coating Fundamentals Symposium.

[‡] Corresponding authors (david.vidal@fpinnovations.ca and francois.bertrand@polymtl.ca).

increasing polydispersity was observed. Finally, following a careful analysis of the inherent numerical errors, an expression relating the Kozeny "constant" to the size distribution and compression level was derived from the simulation results, which led to a modified correlation.

Keywords: Porous Media, Permeability, Particle Size Distribution, Compression, Lattice Boltzmann Method, Parallel Computing

6.1 INTRODUCTION

Porous media are undoubtedly among the most complex structures found in nature. Their complexity arises from the large local variations in pore size and connectivity, which results in transport phenomena through these porous media that may involve a wide range of length scales and speeds. The understanding of transport phenomena in porous media, and more specifically of fluid flow through porous media, is of great scientific and technological importance for many fields of research. This is the case for instance in paper coating, where the porous structure of the coating applied onto the basesheet aims at improving the mechanical, barrier and optical properties as well as the printability of the coated paper. The formulation of the so-called coating colors from the wide variety of available pigments and binders is usually done empirically or guided by experience using laboratory and pilot coater experimentation. This is a costly and time-consuming undertaking that generally does not lead to an optimal formulation and does not give much insight into why one formulation performs better than another. Over the years, for the sake of better product quality and the design of new products, paper coating researchers have gained knowledge on how the end-use properties of coated papers and the on-machine runnability of coating colors relate to the dry and wet coating structures. Consequently, the interest for a fundamental understanding and a more

accurate prediction of the development of coating structures has encouraged several researchers to elaborate new mathematical models based, for instance, on Monte-Carlo techniques (MC), the Discrete Element Method (DEM) or Stokesian Dynamics (Leskinen, 1987; Toivakka *et al.*, 1992, 1997; Eksi and Bousfield, 1997; Toivakka and Nyfors, 2001; Vidal *et al.*, 2003a, 2003b, 2004; Hiorns and Nesbitt, 2003; Bertrand *et al*, 2004; Lyons and Iyer, 2004; Desaulniers *et al.*, 2005; Sand *et al.*, 2006; Alam *et al.*, 2007). The reader is referred to Vidal and Bertrand (2006) for a comprehensive literature review on this topic.

Despite the progress that has been made concerning the modeling of pigment deposition and compression, work on the validation of numerical pigment packings using experimental data has been somewhat limited (Vidal and Bertrand, 2006). Because it is easy to measure experimentally (e.g. by mercury intrusion) and evaluate numerically, packing porosity has generally been the sole parameter used for validation purposes. However, it is well known that packings with completely different structures (e.g. different tortuosities) may have similar overall porosities, which means that a given porosity is not necessarily related to a unique packing structure. Porosity alone is therefore insufficient for characterizing packing structures.

Thorough validation requires the development of advanced numerical pore space characterization tools. This is a challenging task because of the relative complexity of the underlying algorithms and the amount of computations involved. For example, numerical pore size distribution and permeability measurements require fine digitization of the porous medium, which translates into large memory requirements and high computational cost. To alleviate this problem, the use of large distributed memory parallel computers is crucial.

Toivakka and Nyfors (2001) developed a method, recently adapted by Desaulniers (2003) to packings of ellipsoids, that characterizes pore size distribution through a digitized erosion-dilation process of the pore space. The method also gives other information such as surface area, pore connectivity and fractal dimension. Note that

pore size distributions obtained by this method cannot be directly compared to standard mercury intrusion measurements because of different pore definitions. Furthermore, when the structures to be analyzed become highly anisotropic, pore size may become less meaningful because of the elongation of some pores.

Fluid permeability represents a much better packing structure descriptor since it is much more sensitive to structure differences than porosity, and is uniquely defined at low Reynolds number (Re < 1) through Darcy's equation (e.g. Dullien, 1979):

$$\mathbf{v} = -\frac{\mathbf{K}\cdot\nabla P}{\mu}, \tag{6.1}$$

where $\mathbf{v}$ is the superficial velocity, μ the fluid viscosity, ∇P the pressure gradient and $\mathbf{K}$ the second-order permeability tensor with $\mathbf{K} = k\boldsymbol{\delta}$ for homogenous isotropic porous media. In the definition of $\mathbf{K}$, $\boldsymbol{\delta}$ is the identity tensor and k the fluid permeability constant. Note that fluid permeability is not only a good porous structure descriptor, it is also a material property that plays a leading role in air, water or ink solvent uptake. As a result, it affects various end-use properties of coated papers.

Numerous experimental attempts to relate fluid permeability to porosity have been made over the last decades (Bear, 1972; Dullien, 1979; Petrasch *et al.*, 2008). These have led to semi-heuristic or empirical correlations, among which the Carman-Kozeny correlation is the most widely used for $\varepsilon \leq 50\%$ (Bear, 1972; Dullien, 1979):

$$k = \frac{1}{c_K}\frac{1}{S_o^2}\frac{\varepsilon^3}{(1-\varepsilon)^2}, \tag{6.2}$$

where the Kozeny constant is defined as:

$$c_K = c_o\tau^2 = c_o\left(\frac{L_{eff}}{L}\right)^2. \tag{6.3}$$

In these expressions, ε is the effective packing porosity, S_o the pigment specific surface area based on the volume of solids, c_o a pore shape factor and τ the tortuosity, i.e. the ratio of the average effective flow path length (L_{eff}) to the sample thickness in the direction of the flow (L). For mono-sized packings of spheres, we have $\tau = \pi/2$ and $c_o \approx 2$, so that $c_K \approx 5$. For polydisperse packings of spheres with an effective mean diameter D_{eff}, the information about the particle diameter probability distribution function (PDF) p(D) as well as the particle shape is embedded in the pigment specific surface area. For a spherical pigment, this can be written as:

$$S_o = \frac{\int_0^\infty D^2 p(D) dD}{\frac{1}{6}\int_0^\infty D^3 p(D) dD} = \frac{6}{D_{eff}}. \qquad (6.4)$$

Dullien (1979) reported that, in the case of packings made of "particles that deviate strongly from the spherical shape, with broad particle size distributions, and consolidated media, the Carman-Kozeny correlation is often not valid, and, therefore, it should always be applied with great caution" despite the fact that it has proven valuable in many circumstances (e.g. Petrasch *et al.*, 2008).

Several researchers have applied various indirect numerical methods and direct computational fluids dynamics (CFD) methods to study more systematically fluid permeability or pore structure imbibition. Among the indirect methods, we can cite: 1) the resistance network models where the actual pore structure is approximated by a simplified network of interconnected pore bodies and throats to which pore volumes and flow resistances are assigned according to actual pore size distribution measurements and Poiseuille flow approximations (Singh and Mohanty, 2003; Bousfield and Karles, 2004), and 2) the singularity and Oseen equation-based approaches where permeability is deduced from approximations of the hydrodynamic forces acting on collections of spheres in Stokes regime (Clague and Phillips, 1997; Ladd, 1990).

The more direct and accurate approach that consists of solving the Navier-Stokes or Stokes equations within the porous structure using CFD methods has been considered

rather out of reach for decades due to the complexity of the problem and has been limited to small assemblies of mono-sized spherical particles (e.g. Xu and Jiang, 2008). With the recent development of the Lattice Boltzmann Method (LBM) as an efficient way to simulate fluid flow through porous media (see the large body of work recently published in the area, e.g. Bernsdorf *et al.*, 2000; Clague *et al.*, 2000; Hill *et al.*, 2001 Pan *et al.*, 2001; Zeiser *et al.*, 2001; Humby *et al.*, 2002; Kang *et al.*, 2002; Manwart *et al.*, 2002; Tolke *et al.*, 2002; Zeiser *et al.*, 2002; Hayashi *et al.*, 2003; Aaltosalmi *et al.*, 2004; Belov *et al.*, 2004; Guodong *et al.*, 2004; Quispe and Toledo, 2004; Tang *et al.*, 2005; Van der Hoef *et al.*, 2005; Fredrich *et al.*, 2006; Jia and Williams, 2006; Pan *et al.*, 2006; Selomulya *et al.*, 2006; Sullivan *et al.*, 2006; Yamamoto and Takada, 2006; Sullivan *et al.*, 2007; Hayashi and Kubo, 2008; Videla *et al.*, 2008) and the availability of high-performance computing clusters, it is now possible to directly investigate realistic porous systems and thus assess the extent of validity of empirical models such as the Carman-Kozeny correlation.

Unlike the traditional CFD methods that solve directly the Navier-Stokes equations, LBM actually "simulates" macroscopic flows by means of a particulate approach consisting in iteratively streaming and colliding populations of fictitious particles over a discrete lattice grid according to some precisely defined rules. When compared to traditional CFD methods, which have proven limited for solving the Navier-Stokes equations in porous media, LBM is advantageous in three aspects: 1) its relative ease of implementation, 2) its flexibility in discretizing complex geometries by means of a simple structured lattice on which the fluid and solid phases are encoded in a Boolean manner, and 3) the inherent locality of its scheme, which makes it straightforwardly suitable for parallelization on distributed computers. These properties allow LBM to tackle large complex computational domains such as the pore space of polydisperse pigment packings.

Despite the significant advances made possible by LBM, there is still, to our knowledge, a rather limited body of work that has investigated fluid flow through porous media made of *highly* polydisperse particles. Van der Hoef *et al.* (2005) used LBM to study fluid flow through random mono- and bidisperse arrays of spheres, covering a range of porosities from 36% (dense packings) to 90-99% (dilute suspensions). Using data from Ladd (1990) and Hill *et al.* (2001), they proposed a semi-heuristic correlation for the permeability of monodisperse packings/suspensions of spheres of diameter D_{mono}, extending the validity of Carman-Kozeny expression to lower solid contents ($\varepsilon > 50\%$):

$$k_{mono} = D_{mono}^2 \left[180 \frac{(1-\varepsilon)^2}{\varepsilon^3} + 18\varepsilon(1-\varepsilon)\left(1+1.5\sqrt{(1-\varepsilon)}\right) \right]^{-1} . \tag{6.5}$$

They also proposed an expression for polydisperse systems, based on an extrapolation from LBM mono- and bidisperse data, which was not validated by simulations or experiments:

$$k_{poly} = \frac{k_{mono}}{D_{mono}^2} \langle D \rangle^2 \left[1 - 0.064\varepsilon \sum_{i=1}^{n} \left(x_i \frac{D_i}{\langle D \rangle} \right) \right]^{-1} , \tag{6.6}$$

where i denotes the i^{th} class of particles with diameter D_i among n classes of particle sizes, x_i the corresponding solid volume fraction and $\langle D \rangle$ a weighted harmonic mean diameter defined as

$$\langle D \rangle = \left[\sum_{i=1}^{n} \frac{x_i}{D_i} \right]^{-1} . \tag{6.7}$$

Note that the limiting case of n=1 in Equation (6.6) does not yield Equation (6.5).

The objective of this work is threefold. First, to assess the accuracy of LBM for the evaluation of the fluid permeability of *highly* polydisperse pigment packings, i.e. packings made of pigments with realistic particle size distributions (PSD). Second, to compare the properties of simulated packings obtained by means of Monte-Carlo

techniques to those of real commercial pigment packings. Third, to investigate numerically the impact of pigment PSD and packing compression on fluid permeability, and compare the numerical results to those predicted by the Carman-Kozeny correlation and that of Van der Hoef *et al.* (2005). In particular, it will be shown how an expression relating the so-called Kozeny "constant" to the PSD and compression level can be derived from the simulation results after a careful analysis of the numerical errors. Finally, it is the first time, to our knowledge, that such large-scale fluid flow simulations for packings of pigments with realistic particle size distributions were performed and analyzed.

6.2 METHODOLOGY

6.2.1 Experimental

Three isometric natural ground calcium carbonate (GCC) pigments from Omya AG were used for the validation of our MC/LBM simulation model described below: a coarse broad particle size distribution pigment (Hydrocarb 60: GCC-CB), a fine broad PSD pigment (Setacarb: GCC-FB) and a narrow PSD pigment (Covercarb 75: GCC-N). Figure 6.1 displays a scanning electron microscope (SEM) picture of GCC-CB that points out the isometric nature of the pigments. Figure 6.2 gives the PSDs of the three GCC used (continuous lines), measured by a sedimentation-based particle size analyser (SediGraph™ 5100 from Micromeritics Instrument Corporation), and the discretized PSDs (open symbols) as explained in Section 6.2.2.1. Table 6.1 lists their respective specific surface area values, measured by Brunauer-Emmett-Teller (BET) adsorption ($S_{o,BET}$), and, assuming that these pigments are smooth spherical particles, calculated from Equation (6.4) ($S_{o,PSD}$).

Figure 6.1 – SEM picture of ground calcium carbonate (GCC-CB).

Table 6.1 – Specific surface areas of the three GCCs used, evaluated by BET adsorption ($S_{o,BET}$) and by the PSD discretization ($S_{o,PSD}$) according to Equation (6.4).

Pigment	$S_{o,BET}$ (m^2/g)	$S_{o,PSD}$ (m^2/g)
GCC-CB	7.0	3.3
GCC-N	9.0	4.1
GCC-FB	19.2	10.4

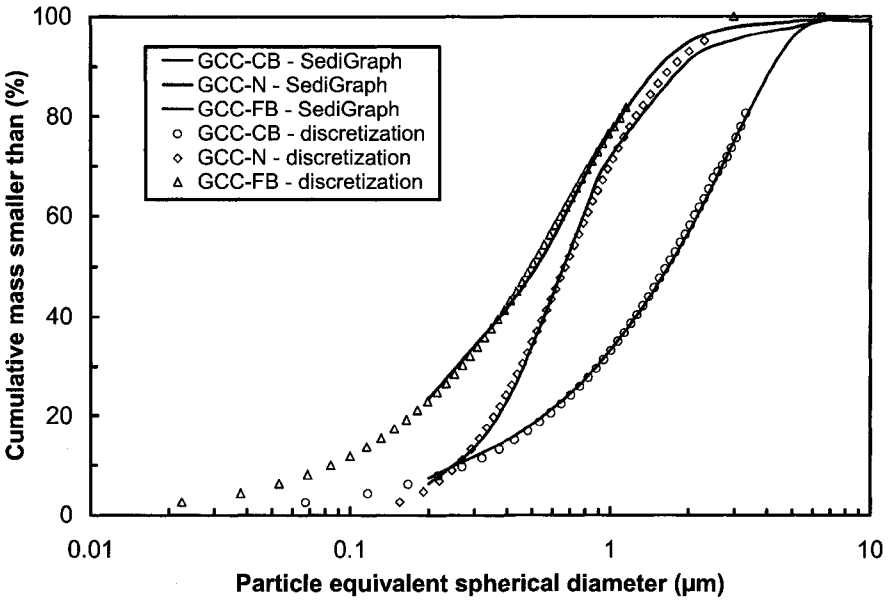


Figure 6.2 – PSDs of the three GCCs used for validation purposes, measured by SediGraph™ 5100 (continuous lines), and their respective discretizations (open symbols).

For the particle size measurements with the SediGraph™ 5100, 10.0 g of mineral powder was mixed with 100 cm^3 of a 0.1 w/w% solution of tetra-sodium polyphosphate (NaPoli) using a high speed blender Polytron PT 3100 from KINEMATICA AG for 3 minutes. The solution used acts as a dispersant for the pigments. It is known not to flocculate pre-dispersed pigments in both dilute and concentrated suspensions, and provides dispersion for non pre-dispersed pigments. Each sample was placed in an ultrasonic bath for 10 min prior to measurement.

In the case of BET adsorption, the condensation isotherm of nitrogen gas onto the surface of the skeletal material was used to evaluate the accessible surface area. The specific surface area was determined using the Tristar 3000 from Micromeritics Instrument Corporation with SmartPrep sample preparation, and a degassing time of 30 min at 250°C.

The pigment powder was equilibrated in an atmosphere of 100% relative humidity at 23°C prior to tablet formation. Then, 60.0 g of homogenised GCC powder was compacted in a hydraulic press for 5 minutes at a predetermined pressure. The hydraulic press consists of a cylindrical hardened steel die, attached to a baseplate with a single acting upper piston. The die can be divided into two parts to aid removal of the compacted pigment sample. The walls of the die were protected with a strip of plastic film to prevent powder sticking and reduce edge friction.

To evaluate sample porosity, mercury intrusion data were obtained from an Autopore III porosimeter from Micromeritics Instrument Corporation using the technique described in Gane *et al.* (1996) up to an applied pressure of 415 MPa. With this method, the intrusion data are corrected using spreadsheet-based program Pore-Comp (from the Porous Media Research Group at the University of Plymouth, U.K.), which uses a blank run correction with the Tait equation (Cook and Hover, 1993) to correct for mercury compressibility and penetrometer expansion effects. The procedure is described in Gane *et al.* (2000).

Liquid permeability was determined using a custom-made cell design, as explained in Ridgway *et al.* (2003). In this technique, gas overpressure is supplied to the permeating liquid from a nitrogen bottle and passes a precision pressure reduction valve (Messer FM 62). A Y-piece connects a digital barometer (Eurolec 0 - 7 bar) to the pressure line. The pressure cell is fixed on a tripod over a micro balance. A PC samples the balance data and records the flux of liquid through the sample. Cycles of measurements are subsequently performed first with the highest possible pressure ($\approx$ 7 bar) and then recorded in descending order of pressure. A decreasing series of pressure steps are used to record the permeation flow over a reasonable amount of time to achieve a curve with a usable gradient. Each step in the recorded curve relates to the occurrence of one drop falling into the weighing pan. By making a linear regression analysis, a gradient is determined, which represents a flow rate of mass per unit time. In the current work, the r^2 values obtained for the linear regression varied between 0.991 and 0.996.

The repeatability of the measurement was also excellent. Therefore, the main source of experimental error (which can be appreciated by the data dispersion for sample GCC-CB in Figure 6.6 at porosity around 26-27%) came from the sample preparation itself.

6.2.2 Computational

The numerical simulations performed in this work require three steps: 1) the discretization of the pigment PSDs, 2) the generation of modeled packings using MC and 3), the fluid flow simulations using LBM. Each of these three steps is discussed in detail.

6.2.2.1 PSD discretization

In this work, pigment particles were assumed to be spherical. This is considered as a reasonable approximation owing to the isometric nature of the GCC pigments used in the experiments. The PSDs of the pigments were discretized using 45 different sizes spread across a wide size range to fit both the actual PSD and the computational domain size. Attention was paid to the smallest particle size used in the discretization since it can affect significantly the specific surface area: the first size was chosen to represent 2.5% of the overall volume. Since the term $1/S_o^2$ appears in the Carman-Kozeny correlation (Equation (6.2)), this can have a significant impact on permeability and thus explain why significant errors can be made when computing it. Figure 6.2 gives the PSD discretization of the 3 GCCs used for validation purposes. Since no SediGraph™ 5100 data were available below 0.2 μm, extrapolations of the PSD curves were necessary to obtain appropriate discretizations. For this purpose, several common distribution functions were tested and, for each pigment, the one providing the best fit with the available

experimental data was selected to represent the full pigment PSD. The GCC-N PSD curve was best fitted with a lognormal distribution whereas, for the other two GCCs (GCC-CB and GCC-FB), a Weibull distribution was found to be better suited. The choice of the distribution function is quite important because it determines the small particle fraction, which may affect the pigment specific surface area and the permeability.

For the lognormal size distribution, the pigment cumulative mass fraction M is related to the particle diameter D as:

$$M(D) = \frac{1}{2}\left(1 + erf\left[\frac{\ln\left(\frac{D}{D_{med}}\right)}{\sqrt{2}\ln\sigma}\right]\right). \tag{6.8}$$

For the Weibull size distribution (also known as Rosin-Rammler-Bennett distribution (Perry and Green, 1984)), this relationship is:

$$M(D) = 1 - e^{-\left(\frac{D}{D_W}\right)^n}, \tag{6.9}$$

where D_{med} is the median particle diameter, D_W the particle diameter at ~63.2 wt%, σ the geometric standard deviation ($\sigma > 1$) and n the power ($n > 0$), these latter two parameters defining the spread of the corresponding distributions: the higher the spread σ or the lower the power n, the more polydisperse the size distribution, as depicted in Figure 6.3. Also, for similar slopes around the same median particle diameter, a Weibull PSD has a higher ratio of fine to large particles than a lognormal PSD. Note that spreads $\sigma = 1$ or $n = \infty$ represent monodisperse particulate systems. Table 6.2 displays the parameters of the PSD models for the three GCCs tested in this work.

Table 6.2 – Best PSD models and their parameters for the three GCCs used in this work.

Pigment	Best PSD model	D_{med} or D_W (μm)	σ or n (-)
GCC-CB	Weibull	2.20	1.16
GCC-N	lognormal	0.671	2.12
GCC-FB	Weibull	0.715	1.06

Note that, to investigate numerically the effect of the pigment polydispersity and packing compression on fluid permeability, both lognormal and Weibull distributions with a 0.6 μm median diameter and increasing spreads from monodisperse to highly polydisperse distributions were also used. Typical spread values for commercial pigments range between 2.0 and 3.0 for σ, and between 1.7 and 1.0 for n. In this study, σ values as high as 3.5 and n values as low as 1.0 were investigated.

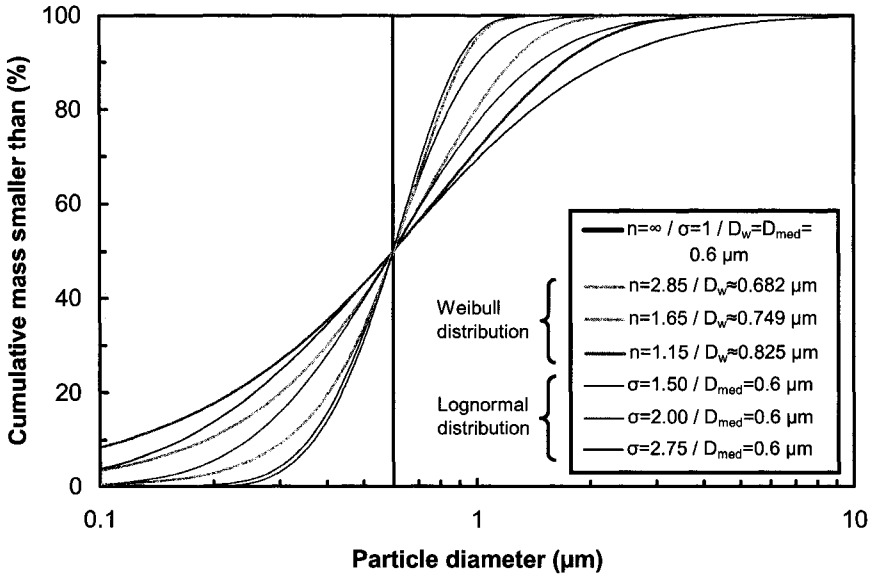


Figure 6.3 – Comparison of lognormal and Weibull PSDs for particulate systems with a median particle size of 0.6 μm.

6.2.2.2 Packing generation with Monte-Carlo methods

Two MC-based methods were considered for the generation of packings. The one that was used the most is a particle filling procedure, which consists of sequentially selecting particles of a given discretized particle size distribution in descending order of size and randomly positioning them within a computational domain of given height so that they do not overlap with already set particles. The positioning of each particle is attempted a number of times defined by the n_{trial} parameter. The computational domain has periodic boundary conditions in all three directions and fixed in-plane dimensions (i.e. dimensions perpendicular to the main flow direction). The smallest domain height leading to the successful positioning of all particles of the PSD discretization for a given n_{trial} value is determined by a bisection search-based algorithm presented in Appendix (Section 6.7). It can be noticed that, by varying the n_{trial} parameter, "compression" levels (i.e. heights and corresponding porosities) similar to those found in the experiments can be obtained. The higher the n_{trial} value, the lower the porosity obtained. For the study of the effect of packing compression on permeability, low, medium and high compression levels were arbitrarily defined; they correspond to n_{trial} values of 3×10^3, 5×10^4 and 1×10^6, respectively. Note that the packing procedure is repeated n_{sim} times for each height and that 50% of the packing attempts must complete for a specific height to be accepted. In this work, n_{sim}=99 and all the packings were simulated in parallel on a computer cluster using a Fortran/MPI implementation. We will refer to this domain filling algorithm as MCP for MC Packing algorithm. The packings used for validation purposes with the three GCC tablets (GCC-CB, GCC-N and GCC-FB) were obtained by means of this packing algorithm.

The second method used in this work is the so-called MC Deposition (MCD) algorithm (Vidal *et al.*, 2003) that consolidates using a gravity-based random walk procedure an initial set of non-overlapping particles within a domain of known dimensions. The resulting packing is then further compressed using a DEM-based

compression scheme and the use of a downward moving wall as described in Pianet *et al.* (2008). This procedure, which is hereafter referred to as MCD/DEM, was used to assess the impact of compression and compression methods on permeability.

Table 6.3 – MC/LBM simulation parameters.

Physical parameters		
Fluid density (kg/m^3):		10^3
Fluid viscosity (Pa.s):		10^{-3}
Pressure drop (Pa):		1
Initial velocity (m/s):		0
Pigment coat weight (g/m^2):	- lognormal distribution	20
	- Weibull distribution	10 (except for GCC-FB: 8.5)
Domain size (μm^2):	- lognormal distribution	20×20 (except for $\sigma \geq 3.00$: 12×12)
	- Weibull distribution	10×10 (except for GCC-FB: 5×5)
Numerical parameters		
Rest population weight w_0:		2/9
Dimensionless relaxation time τ^*:		1
Convergence criterion $((\mathbf{du}/dt)/(\mathbf{du}/dt)_{max})$:		10^{-5}
Lattice size δ_x (μm):	- MCP	$2.50 \times 10^{-2} \geq \delta_x \geq 7.50 \times 10^{-3}$
	- MCD/DEM	4.00×10^{-2}
	- GCC-CB	8.00×10^{-3}
	- GCC-N	1.25×10^{-2}
	- GCC-FB	3.33×10^{-3}

6.2.2.3 Lattice Boltzmann method and fluid flow simulations

LBM is based on the discretization in space (**x**), velocity (**e**) and time (t) of the Boltzmann equation of the kinetic gas theory that describes the evolution of the probability distribution function (or population) of particles, $f(\mathbf{x},\mathbf{e},t)$, according to their microdynamic interactions.

In practice, the populations of particles propagate and collide at every time step δ_t on a lattice with spacing δ_x and along $\mathbf{e}_i$ velocity directions, where the number of directions i (n_d) depends on the type of lattice chosen. A D3Q15 lattice is used in the present work, i.e. a 3-dimensional lattice with n_d=15 velocity directions[43]. The collision-propagation procedure can be mathematically summarized by a two-step scheme comprising a collision step:

$$f_i^*(\mathbf{x},t) = f_i(\mathbf{x},t) - \frac{f_i(\mathbf{x},t) - f_i^{eq}(\mathbf{x},t)}{\tau^*}, \tag{6.10}$$

followed by a propagation step:

$$f_i(\mathbf{x} + \mathbf{e}_i\delta_t, t + \delta_t) = f_i^*(\mathbf{x},t), \tag{6.11}$$

where $f_i(\mathbf{x},t)$ is the particle population in the direction of the velocity $\mathbf{e}_i$ at position $\mathbf{x}$ and time t, and τ^* is a dimensionless relaxation time. The second term of the right-hand side of the Equation (6.10) approximates the collision by means of a single relaxation procedure, the so-called Bhatnager, Gross and Krook (BGK) approximation (Succi, 2001), where the local equilibrium population, $f_i^{eq}(\mathbf{x},t)$, is given for a D3Q15 lattice by:

$$f_i^{eq}(\mathbf{x},t) = w_i\rho\left[1 + 3(\mathbf{e}_i\cdot\mathbf{u})\left(\frac{\delta_t}{\delta_x}\right)^2 + \frac{9}{2}(\mathbf{e}_i\cdot\mathbf{u})^2\left(\frac{\delta_t}{\delta_x}\right)^4 - \frac{3}{2}(\mathbf{u}\cdot\mathbf{u})\left(\frac{\delta_t}{\delta_x}\right)^2\right], \tag{6.12}$$

with $w_0 = \frac{2}{9}$, $w_i = \frac{1}{9}$, for i = 1 to 6 and $w_i = \frac{1}{72}$, for i = 7 to 14.

The dimensionless relaxation time τ^* is related to the kinematic viscosity of the fluid ν by:

$$\tau^* = \frac{\nu}{\delta_t c_s^2} + \frac{1}{2}, \tag{6.13}$$

[43] Defined as $\mathbf{e}_0$=(0,0,0), $\mathbf{e}_1$=(c,0,0), $\mathbf{e}_2$=(-c,0,0), $\mathbf{e}_3$=(0,c,0), $\mathbf{e}_4$=(0,-c,0), $\mathbf{e}_5$=(0,0,c), $\mathbf{e}_6$=(0,0,-c), $\mathbf{e}_7$=(c,c,c), $\mathbf{e}_8$=(-c,-c,-c), $\mathbf{e}_9$=(c,c,-c), $\mathbf{e}_{10}$=(-c,-c,c), $\mathbf{e}_{11}$=(c,-c,c), $\mathbf{e}_{12}$=(-c,c,-c), $\mathbf{e}_{13}$=(c,-c,-c), $\mathbf{e}_{14}$=(-c,c,c), with $c=\delta_x/\delta_t$.

where c_s is the speed of sound of the lattice defined as :

$$c_s = \sqrt{\frac{3(1-w_0)}{7}}\frac{\delta_x}{\delta_t}. \tag{6.14}$$

In our LBM code, the pressure drop is implemented through the use of a body force and periodic boundary conditions in the flow direction. In practice, for better accuracy, τ^* is chosen equal to 1.0 and δ_t and δ_x are chosen according to Equation (6.13). Then, from initial conditions (here, $\mathbf{u}$=0) and appropriate boundary conditions (here, periodic boundary conditions for the outer part of the domain and half-way bounce-back conditions for the inner solid walls), the collision-propagation scheme is marched in time until an appropriate convergence is reached. Finally, the local macroscopic fluid density and velocity can be obtained by:

$$\rho = \rho(\mathbf{x},t) = \sum_i f_i(\mathbf{x},t) \tag{6.15}$$

and

$$\mathbf{u} = \mathbf{u}(\mathbf{x},t) = \frac{1}{\rho}\sum_i f_i \mathbf{e}_i. \tag{6.16}$$

In the limit of low Mach and Knudsen numbers, it is possible to demonstrate using a Chapman-Enskog multiscale expansion that LBM and its underlying scheme yields the transient incompressible Navier-Stokes equations up to a truncation error, the order of which depends on the boundary conditions used (Succi, 2001). The LBM scheme is explicit and the population update at a lattice node is a local operation since it only requires the populations of the immediate neighbouring nodes. This makes the LBM scheme well suited to parallelization on a distributed memory computer. The reader is referred to Succi (2001) and Nourgaliev *et al.* (2002) for more details on LBM and its parallelization.

Fluid flow through the numerical packings described in the previous sections was

predicted by means of massive parallel three-dimensional LBM simulations performed on a High-Performance Computing Cluster (Mammouth(mp)) from the Réseau Québécois de Calcul de Haute Performance (RQCHP) and a newly developed single-vector LBM implementation based on the OpenMPI library. This implementation allows a significant reduction of memory usage and includes a workload balance scheme that takes into account porous media properties. More details will be found in a forthcoming paper. The results obtained with this implementation were found to be second-order accurate in space. Also, in the case of the permeability of hexagonal arrays of cylinders with various diameters, a test case problem used to assess our LBM code, the relative error with respect to the analytical solution was found to be less than 1.4%. Table 6.3 summarizes the physical and numerical parameters for the MC/LBM simulations carried out in this work. Up to 256 3.6 GHz Intel Xeon processors were used for these computations. Overall memory usage for each LBM simulation varied between 50 to 350 GB depending on the problem size (the largest lattice size was 1600 × 1600 × 1328). The overall computational time was between 1 and 12 hours when using 256 processors, which corresponds to an average computational speed of ~$4x10^8$ lattice-site updates per second.

6.3 RESULTS AND DISCUSSION

In this section, the conditions for getting accurate LBM simulations are first discussed. Next, the permeability values obtained from the LBM simulations are compared to experimental data for the GCC-CB, GCC-N and GCC-FB tablets and values obtained from the Carman-Kozeny correlation. Finally, the impact of the particle size distribution spread and packing compression on permeability as predicted by LBM simulations is examined.

Figure 6.4 – Cross-section of the velocity field (in m/s) as computed by LBM through a 46% porosity MCD/DEM packing of low polydispersity pigments (σ=1.5).

6.3.1 Accuracy of MC/LBM in the case of polydisperse packings

An example of the velocity field obtained with LBM is illustrated in Figure 6.4. There are three main sources of numerical uncertainty with respect to these MC/LBM simulations. All error bars in the following sub-sections will reflect the numerical uncertainty due to these three sources of error. The first one is related to the inherent variability of the MC packings formed with a given PSD, resulting in a standard deviation of the permeability which, for our simulations, was found to be lower than 0.7% of the average value. This very low value is due to the large domains used.

The second source is related to the size of the domain itself. Clague and Phillips (1997) used the notion of Brinkman screening length to determine the minimum domain size above which the hydrodynamics no more varies. We have observed this approach is valid for monodisperse or low polydispersity pigments. For high polydispersity pigments,

the domain size needs to be substantially larger than that provided by the Brinkman screening length to ensure an adequate discretization of the whole PSD and more specifically of the coarse particles. The domain size required for high polydispersity pigments was then used as a basis for all simulations. In practice, as the ratio of the domain dimension in the direction of the flow (i.e. h, the height of the packing) to the diameter of the largest particles (D_{big}) increased, a nearly-quadratic monotonically decreasing convergence of the permeability to a plateau was observed. In the simulations, permeabilities less than 1% of the plateau values were obtained by setting $h/D_{big} \geq 2.5$, except for highly polydisperse cases ($\sigma \geq 2.75$ and $n \leq 1.50$) where $2.5 > h/ D_{big} \geq 1.3$ was set and permeabilities less than 7% of the plateau values were achieved.

The third and most obvious source of numerical error is related to lattice spacing and must be treated with care. The difficulty with simulating fluid flow through polydisperse pigment packings stems from the fact that 1) the computational domain is usually large enough to accommodate the largest particles and be representative of the whole PSD as discussed above, and 2) its discretization needs to be fine enough in order to resolve adequately fluid flow in the numerous small pores created by the finest particles. This leads to very large numbers of computational lattices. On the other hand, fluid flows through preferential pathways (the so-called channelling effect) that will be usually larger than these smaller pores. This leads to the following question: how small should the LBM lattice size be?

Figure 6.5 presents the convergence of the solution as the lattice is refined for several MCP packings with various lognormal (σ) and Weibull (n) PSD spreads. As expected, as the lattice size decreases, the relative error decreases. Note that, in all cases, the permeability converges quadratically to a lower value as the lattice is refined. It appears that, as long as the ratio $D_{mean}/(\delta_x (1- \varepsilon)^{1/3})$ is higher than 10, the error on the permeability value is lower than around 10%, which is reasonable. For this reason, all the simulation results subsequently discussed were performed with $9.8 \leq D_{mean}/(\delta_x(1- \varepsilon)^{1/3}) \leq 35.7$ and, as expected, the error usually increased with the polydispersity.

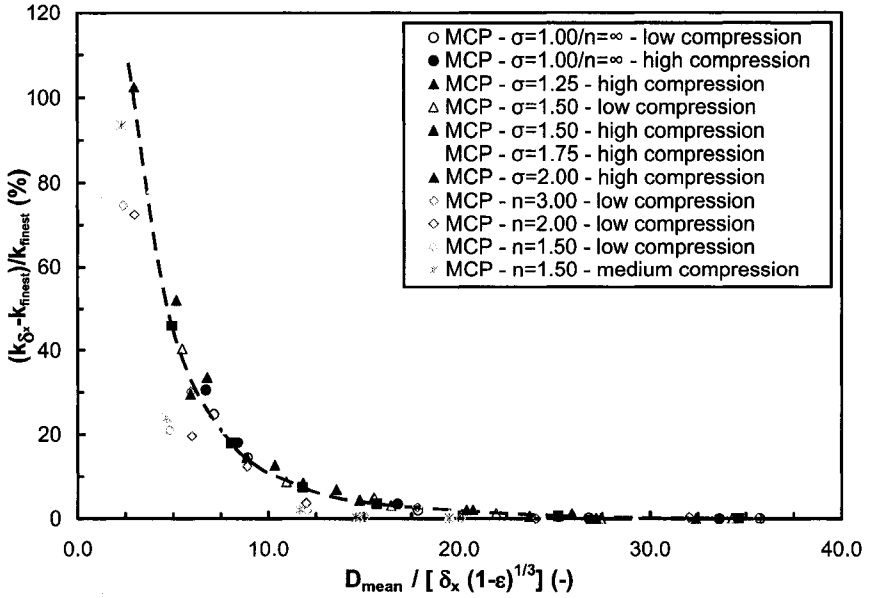


Figure 6.5 – Relative error on the permeability (with respect to the finest lattice simulation) as a function of the ratio of the mean particle diameter (D_{mean}) to the LBM lattice spacing normalized by the one-dimensional packing fraction.

6.3.2 Experimental and analytical validation

To assess the accuracy of the numerical model, the MCP/LBM permeability results were first compared to those of the Carman-Kozeny correlation with c_K=5.00, and to experimental measurements on the GCC tablets described in the previous section. First, it can be readily seen in Figure 6.6 that, considering the well-known sensitivity of permeability results, the agreement between the numerical results and the experimental data is excellent: the average relative error on the experimental values is 31%, 39% and 105% for GCC-CB, GCC-FB and GCC-N, respectively. Previous works from the literature have often reported errors in the range of one to two orders of magnitude. Also, experimental measurements are subject to experimental errors as evidenced, for instance, by the spread of the GCC-CB experimental data for porosities around 26-27%.

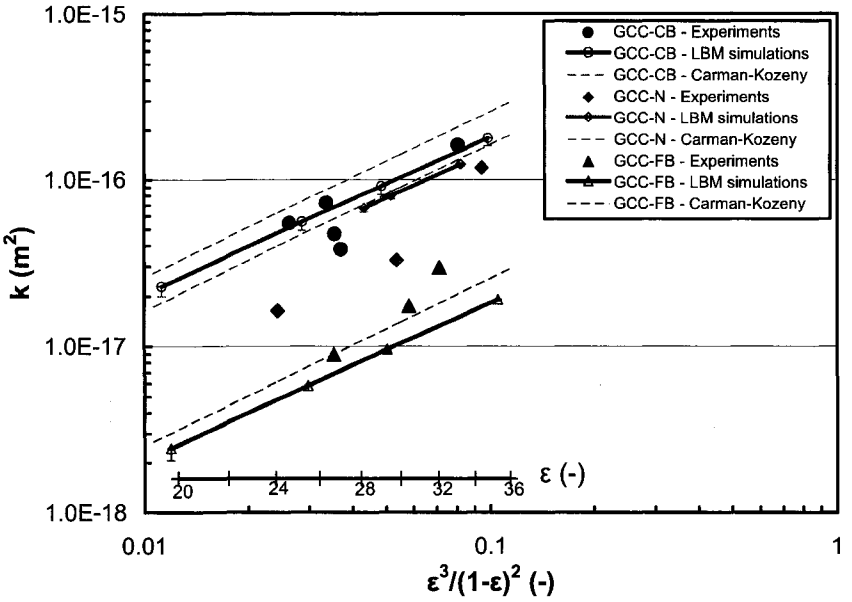


Figure 6.6 – Comparison of MCP/LBM permeability results to experimental and Carman-Kozeny (S_o= $S_{o,PSD}$ and c_K=5.00) permeability values, as a function of porosity for the three GCC pigments.

Surprisingly, the pigment with the narrowest PSD (GCC-N) gives the largest relative error. Note that the PSD for this pigment was fitted with a lognormal distribution instead of the Weibull distribution for the other two pigments. Interestingly, GCC-N packings could not be created with MCP for $\varepsilon < 27\%$ whereas tablets with $\varepsilon \approx 24\%$ could be made. The extrapolation of the PSD below particle size of 0.2 μm may explain this difference.

Overall, there are several likely causes for discrepancies between numerical results and experimental data: 1) the particle is assumed to be spherical, 2) the extrapolation of the tail of the PSD and the PSD measurement itself, 3) the numerical error inherent to the MCP model and 4) the numerical error inherent to LBM. The particle shape no doubt adds to the error. As can be seen in Table 6.1, specific surface area values measured by BET adsorption ($S_{o,BET}$) are roughly twice as large as the ones used in the simulations ($S_{o,PSD}$ values assume smooth spherical particles) for all pigments. This can be explained by both the surface texture of the pigments and the actual particle shape. Also, it was noted that even small variations in the fraction of small particles or, equivalently, truncation of the small diameter tail of a PSD curve at a slightly different particle size, can lead to rather significant variations in the specific surface area and thus

in the permeability values obtained. This may explain why evaluating permeability is prone to significant errors.

Numerical permeability values and experimental data are also compared in Figure 6.6 to the Carman-Kozeny correlation with $S_o = S_{o,PSD}$. Note that $S_{o,BET}$ was also used although the corresponding results are not shown here because of its lower predicting capability (one explanation for this is that $S_{o,BET}$ also accounts for surface texture at scales that do not affect the flow since they are smaller than the laminar fluid boundary layer, as pointed out by Carman (1956)). Good adequacy can be noticed. More precisely, the agreement between the Carman-Kozeny correlation and the LBM results is remarkable for GCC-N, whereas fairly small differences can be observed for the other two pigments. Overall, considering the experimental error and the approximations made on particle shape and PSD, these results demonstrate the suitability of the MCP/LBM combination for creating (MCP) packings with permeability (and thus structure) similar to that of real macroscopic tablets, and for investigating (LBM) flow properties through such packings. It can also be noted that the Carman-Kozeny correlation seems valid for all the pigments and PSDs considered. This aspect as well as the impact of PSD and packing compression on permeability will be further investigated in the next section.

6.3.3 Impact of PSD and packing compression on permeability and Carman-Kozeny constant

As can be seen in Figures 6.7 and 6.8, the permeability of packings created with MCP decreases by close to three orders of magnitude when the spreads of lognormal (σ) and Weibull (n) distributions are varied from 1.00 to 3.50 and ∞ to 1.25, respectively. These figures also show that compressing these packings, which was achieved using MCP, reduces significantly the permeability, as expected. Predictions by the Carman-Kozeny correlation with c_K=5.00 are excellent for $1.25 \leq \sigma \leq 2.25$ (Figure 6.7) and $0.1 \leq$

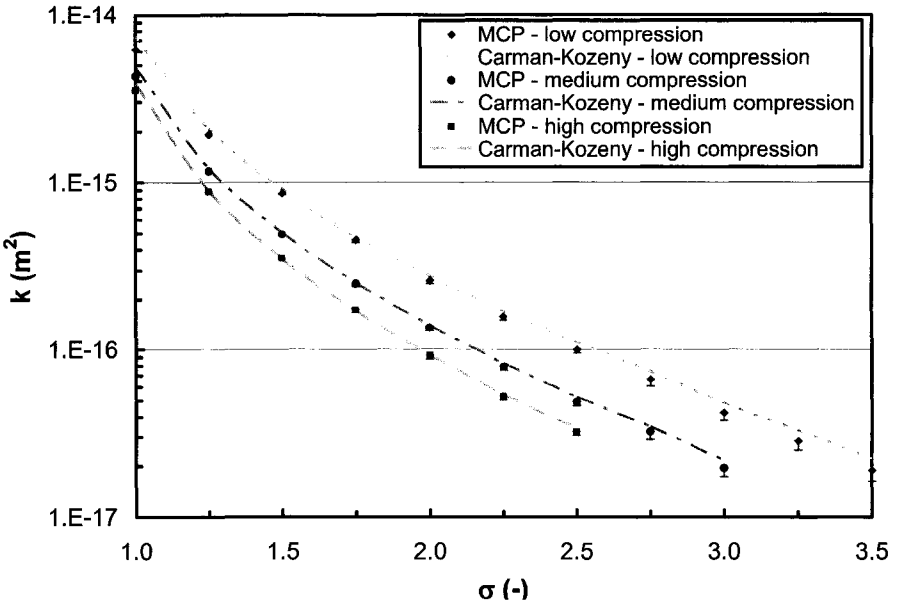


Figure 6.7 – MCP/LBM and Carman-Kozeny permeability predictions as a function of compression and the spread σ of a lognormal PSD (D_{med}=0.6 μm).

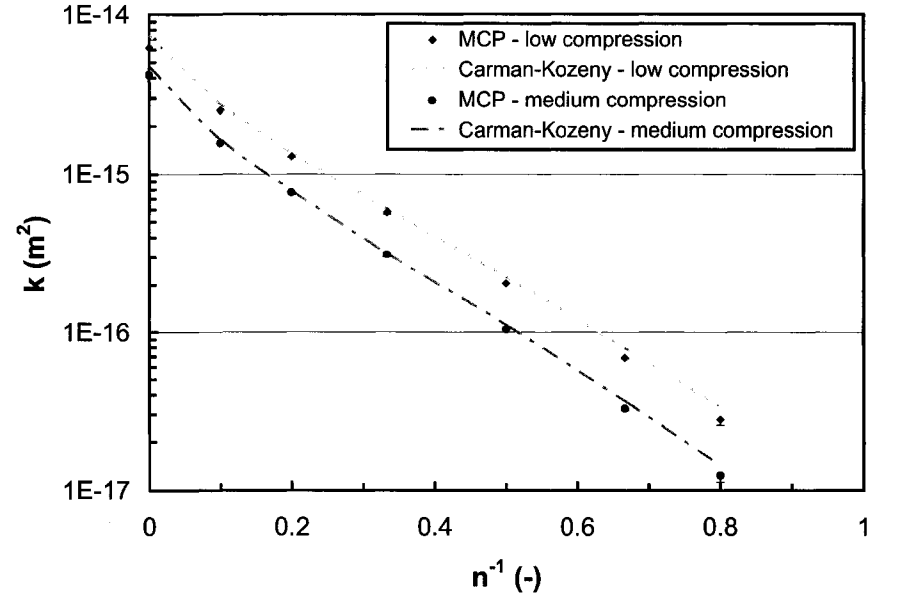


Figure 6.8 – MCP/LBM and Carman-Kozeny permeability predictions as a function of compression and the spread n of a Weibull PSD (D_{med}=0.6 μm).

$n^{-1} \leq 0.4$ (Figure 6.8) for all three compression levels, but the agreement deteriorates when polydispersity is further increased. This can be explained by an increase of the

tortuosity as the PSD becomes wider and the ratio of the number of small particles to the number of large ones is increased. Surprisingly, for the monodisperse cases (σ=1 or n^{-1} =0), the agreement is not as good as expected, especially at low compressions. This is due to the fact that monodisperse packings made with MCP are inherently loose ($\varepsilon > 50\%$) (very loose random packing porosity is ~45%) and thus fall beyond the validity region of the Carman-Kozeny correlation ($\varepsilon \leq 50\%$). As compression is intensified and the packing porosity gets closer to 50%, the agreement is improved.

As already discussed, real packing compressions were simulated using a MCD/DEM procedure. This method consists of first depositing particles with MCD and subsequently compressing them with DEM and the use of a downward moving wall, as explained in Pianet *et al.* (2008). Figure 6.9 illustrates the compression process for σ=1.00 (mono-sized) and σ=1.50. Compression reduces the porosity and the roughness of both packings, the porosity going from that of the very loose random packing (~45%) to close to that of the dense random packing (~36%), in the case of mono-sized pigments. Here again, as the packings are compressed, the permeability values follow generally

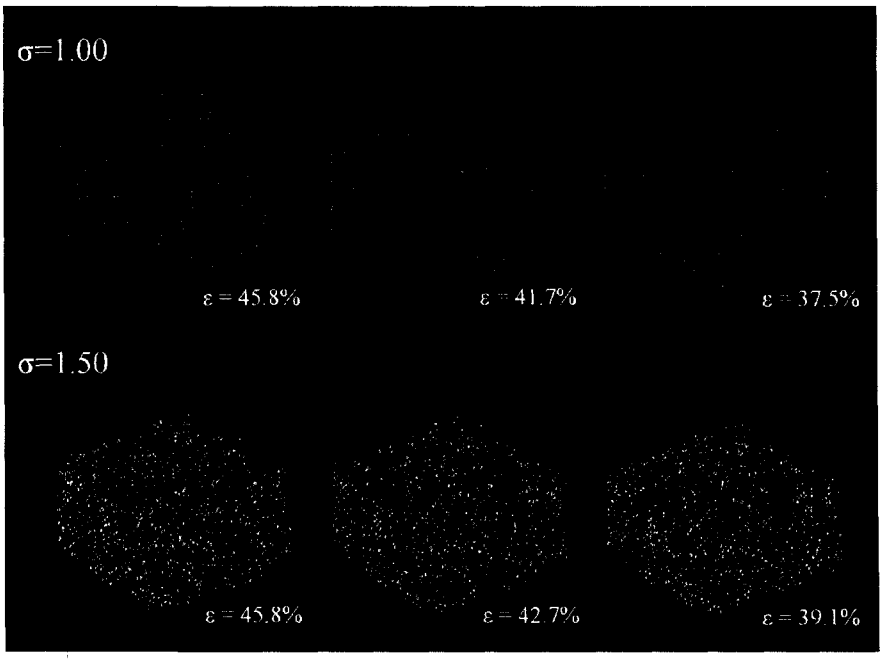


Figure 6.9 – Perspective views of the DEM compression of MCD packings.

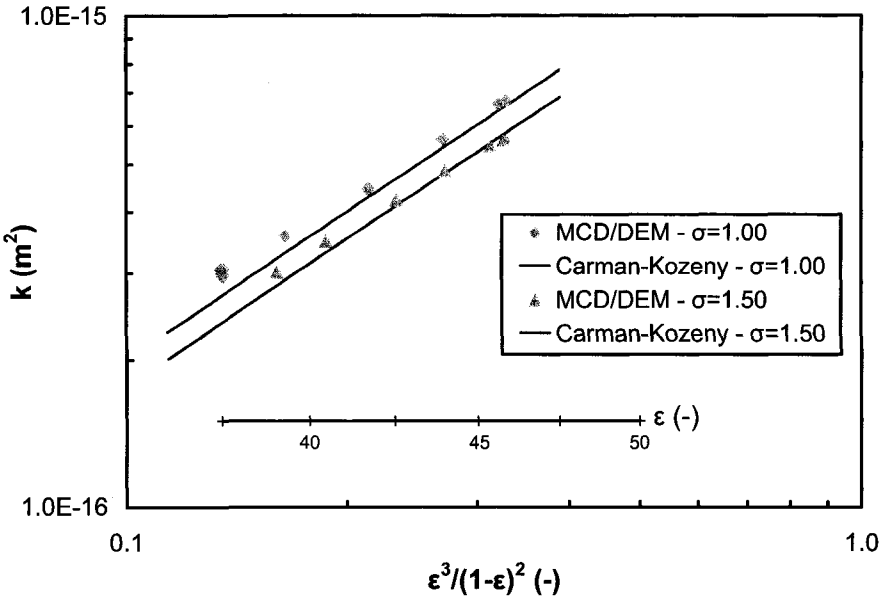


Figure 6.10 – Variation of the MCD/DEM/LBM and Carman-Kozeny (c_K=5.00) permeability values as packings of different spreads σ are compressed and porosity is decreased.

well the Carman-Kozeny correlation with c_K=5.00 (Figure 6.10), although slight deviations at high compression levels (i.e. low porosities) are apparent on this graph. The magnitude of these discrepancies (6-13%) cannot be explained by numerical errors only. Indeed, it indicates that, as compression is intensified, c_K slightly decreases, which means that the permeability predicted is higher than that coming from the Carman-Kozeny correlation with c_K=5.00. We believe this is due to a decrease of the pore shape factor c_0 in Equation (6.3).

The permeability values obtained from all the simulations of this work were normalized with respect to the square of the effective particle diameter (D_{eff}) and plotted against porosity (Figure 6.11). Interestingly, all data points align along one master curve, which follows quite closely the Carman-Kozeny correlation with c_K=5.00. Nevertheless, some deviations from the Carman-Kozeny correlation can be observed if the data are carefully analyzed: 1) the data points for ε > 50% correspond to normalized permeabilities lower than those predicted by the Carman-Kozeny correlation (as explained earlier, this is not surprising as these points are beyond the range of validity of

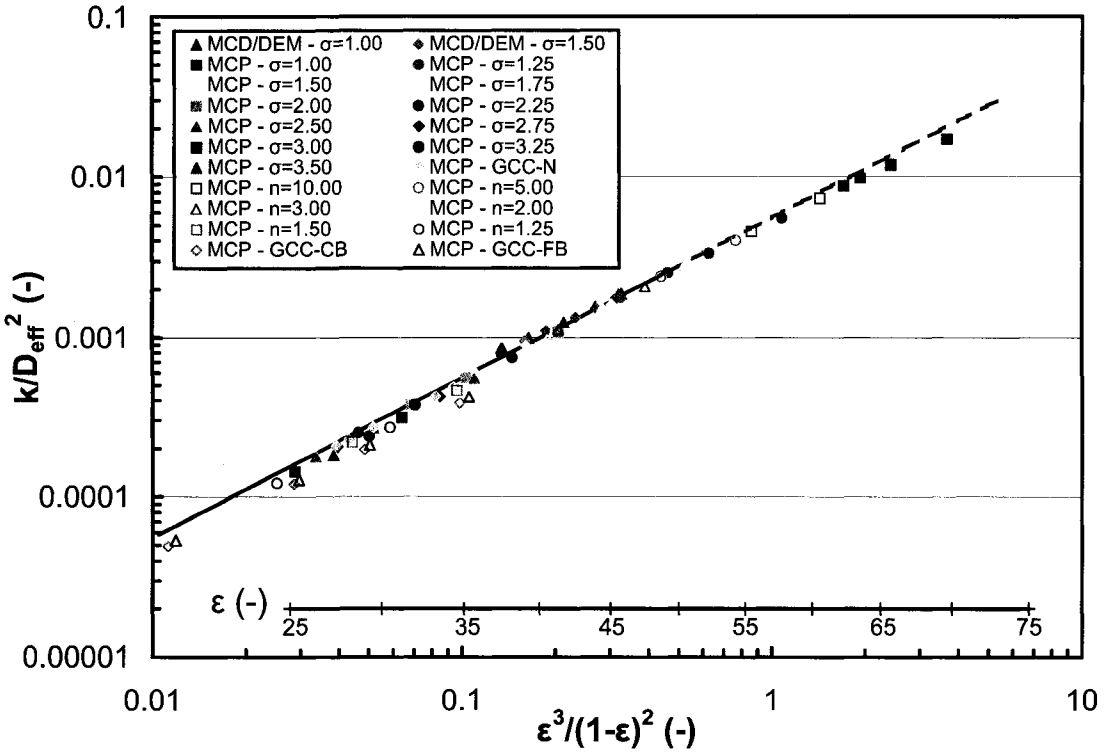


Figure 6.11 – Normalized permeability values as a function of porosity for all simulations. The line represents the Carman-Kozeny correlation with c_K=5.00. The dashed portion is the extension of the correlation out of its limit of validity (ε < 50%). Filled symbols correspond to simulations with lognormal PSD and open symbols with Weibull PSD. Error bars not shown for clarity.

the correlation), and 2) highly polydisperse pigments (σ > 2.50 for a lognormal PSD or n < 2.00 for a Weibull PSD) have permeability values lower than those predicted by the correlation. In the range of polydispersity investigated, the permeability deviation was however relatively low and not higher than 30%. A careful analysis of the numerical errors (see Section 6.3.1) showed that these deviations cannot be attributed to computational artefacts and are thus physical in essence. They can be explained by a higher tortuosity resulting from a higher ratio of small to large particles as polydispersity is increased. As a matter of fact, Figure 6.12 shows that there is a definite correlation between the value of the Kozeny constant and the skewness of the probability distribution function (PDF) defined as

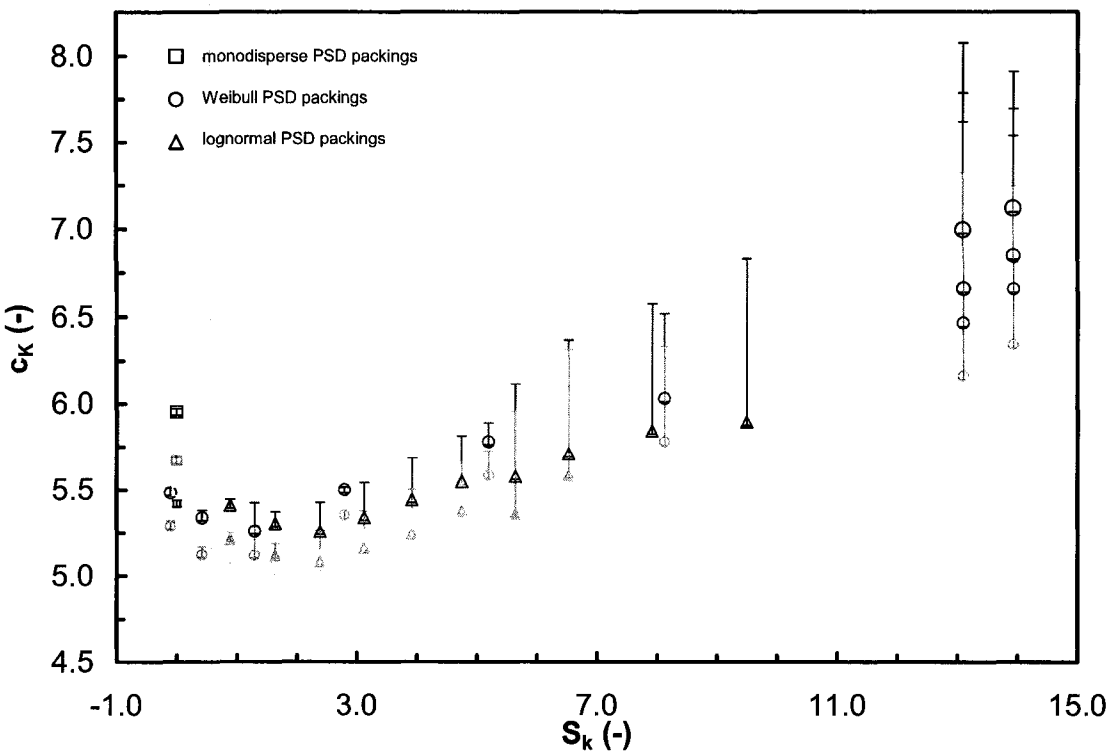


Figure 6.12 – Kozeny constant (c_K) as a function of the particle PDF skewness (S_k) for all the MCP/LBM simulations (except for GCC-N that was removed for clarity). The color scale and the symbol size used for the data points depend on the level of compression: the warmer the color and the smaller the symbol, the higher the compression.

$$S_k = \frac{\int_0^\infty (D - D_{mean})^3 p(D) dD}{\left(\int_0^\infty (D - D_{mean})^2 p(D) dD \right)^{3/2}} \approx \frac{\left(\frac{1}{n} \sum_{i=1}^{n} (D_i - D_{mean})^3 \right)}{\left(\frac{1}{n} \sum_{i=1}^{n} (D_i - D_{mean})^2 \right)^{3/2}}, \tag{6.17}$$

where

$$D_{mean} = \int_0^\infty p(D) D dD \approx \frac{1}{n} \sum_{i=1}^{n} D_i, \tag{6.18}$$

and n is the number of particles. One may also note a clear negative impact of the packing compression on the value of this constant. The range of Kozeny constant values obtained ($4.9 \leq c_K \leq 7.1$) is in good agreement with the range obtained experimentally by Wyllie and Gregory (1955) and numerically by Van der Hoef *et al.* (2005) for both

mono- and bidisperse systems, i.e. $4.8 \leq c_K \leq 7.4$ and $4.9 \leq c_K \leq 10.9$, respectively. Van der Hoef *et al.* (2005) also reported a negative correlation between the Kozeny constant and packing compression. Moreover, it can be observed in Figure 6.13 that, when scaling the Kozeny constant with the one-dimensional packing fraction, all data points for both lognormal and Weibull PSD packings fall, within numerical uncertainty, on a single linear relationship for $-0.1 < S_k < 14.0$:

$$c_K(1-\varepsilon)^{1/3} = c_1 S_k + c_2, \tag{6.19}$$

where $c_1 \approx 0.22$ and $c_2 \approx 4.00$ are fitting constants. The fact that the scaled data do not overlap at high skewness ($13 \leq S_k \leq 14$) may be attributed to a not fine enough lattice

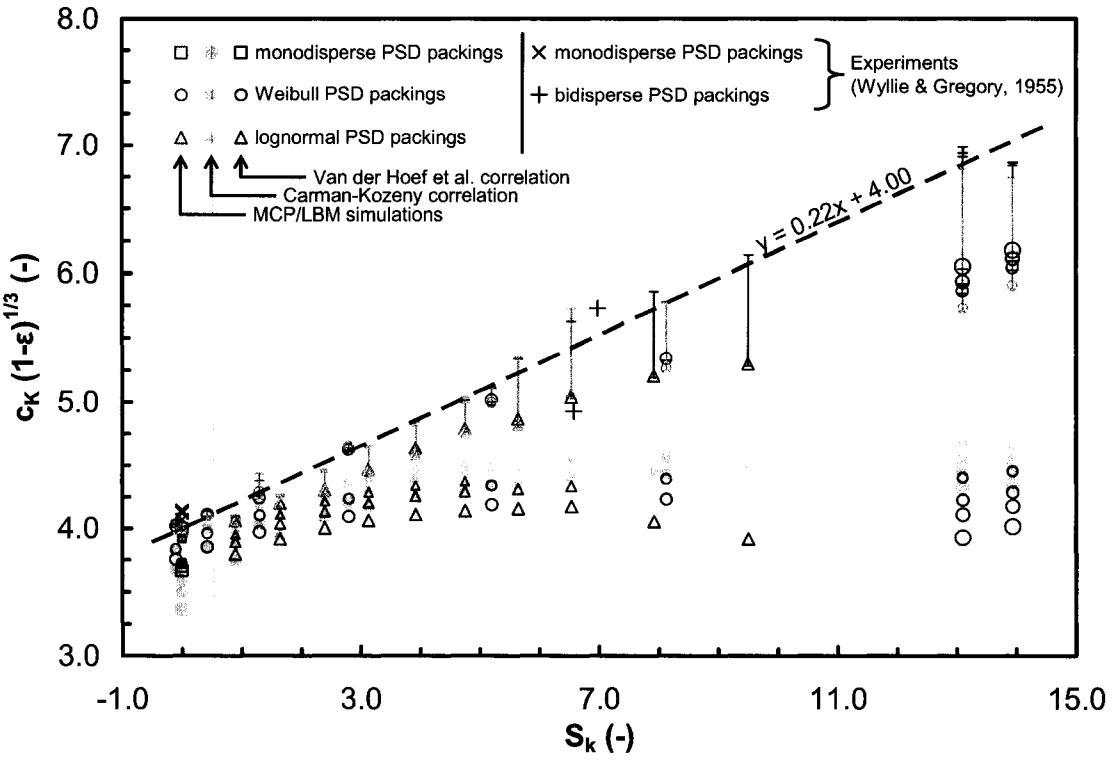


Figure 6.13 – Normalized Kozeny constant ($c_K (1-\varepsilon)^{1/3}$) as a function of the particle PDF skewness (S_k) for all the MCP/LBM simulations (except for GCC-N that was removed for clarity) and as calculated from the MCP packings by Carman-Kozeny (grey filled symbols) and Van der Hoef *et al.* (2005) (black open symbols) correlations. The color scale and the symbol size used for the data points depend on the level of compression: the warmer the color and/or the smaller the symbol, the higher the compression. Experimental data (black crosses) from Wyllie and Gregory (1955) are added for comparison purposes.

resolution ($D_{mean}/(\delta_x(1-\varepsilon)^{1/3}) \approx 10$). Moreover, experimental data from Wyllie and Gregory (1955) for monodisperse and bidisperse packings of spheres comply with the data of Figure 6.13. On the other hand, it can be seen that the predictions from the Carman-Kozeny and Van der Hoef *et al.* (2005) correlations (Equation (6.6)) diverge significantly from our simulation results at high polydispersity (for $S_k > 5$), despite a fair agreement for monodisperse and low polydispersity systems. This result is not surprising because the Van der Hoef *et al.* (2005) correlation is based on an extrapolation from mono and bidisperse simulation results.

Combining Equations (6.2) and (6.19) leads to the following correlation:

$$k = \frac{1}{S_o^2\left(c_1 S_k + c_2\right)}\frac{\varepsilon^3}{(1-\varepsilon)^{5/3}}. \tag{6.20}$$

Contrary to Equation (6.2) that depends on c_K, permeability is now only related to the PSD through the specific surface area and the PDF skewness, and a porosity expression slightly different from that in the original Carman-Kozeny correlation. Our numerical data showed that this proposed correlation is valid for lognormal and Weibull PSDs with $-0.1 < S_k < 14.0$ and monodisperse packings over a wide range of porosities ($\varepsilon \leq 90\%$). In fact, it diverges from Equation (6.5) and our own set of LBM simulation data for monodisperse packings of spheres when $\varepsilon > 90\%$ (not shown here). However, such low solids content systems represent more a dilute suspension than a packing of particles.

6.4 CONCLUDING REMARKS

This work showed that the lattice Boltzmann method (LBM) can be used to measure with good accuracy the fluid permeability of highly polydisperse pigment packings as long as the smallest particles are appropriately discretized. Also, Monte-

Carlo methods appeared to create packings with structures very close to real macroscopic pigment tablets. The agreement with experimental measurements is within ~30-40% for 2 of the 3 pigments that were considered, which is quite reasonable considering the numerical and experimental sources of errors and the well-known extreme sensitivity of permeability results. Interestingly, permeability values obtained from simulations for compressed and mono-sized to highly polydisperse pigment packings proved to be in good agreement with the Carman-Kozeny correlation with c_K=5.00. More precisely, only ~30% lower permeabilities were predicted in the worst cases (i.e. for the highly polydisperse pigments investigated) with both lognormal and Weibull PSDs. Also, as packings were compressed, LBM flow simulations predicted a deviation from the Carman-Kozeny correlation with slightly higher permeability values. In fact, it was shown that the Kozeny "constant" is actually a function of both the pigment PDF skewness and porosity. From this result, a modified Carman-Kozeny correlation valid for both lognormal and Weibull PSD packings was derived. This new correlation is only related to the pigment PSD properties (i.e. through the specific surface area and PDF skewness of the pigment) and a porosity expression slightly different from that in the original Carman-Kozeny correlation. To our knowledge, this is the first time that such relationship is established. Future work will verify the validity of this correlation for other PSD models and non-spherical particle packings.

6.5 ACKNOWLEDGMENTS

The computer resources and support from the Réseau Québécois de Calcul de Haute Performance (RQCHP), and the financial contribution of the NSERC Sentinel Network are gratefully acknowledged.

6.6 REFERENCES

Aaltosalmi, U., Kataja M., Koponen A., Timonen J., Goel A., Lee G., & Ramaswamy S. (2004). Numerical analysis of fluid flow through fibrous porous materials. *Journal of Pulp and Paper Science, 30(9)*, 251-255.

Alam, P., Xu, Q., Toivakka, M., Hämäläinen, H., & Syrjälä, S. (2007). The elastic modulus of paper coating in tension and compression. *In CD-ROM proceedings of the 2007 TAPPI Coating Conference*. Atlanta, GA, USA: TAPPI Press.

Bear, J. (1972). *Dynamics of Fluids in Porous Media*. New York, NY, USA: Dover Publications Inc.

Belov, E. B., Lomov, S. V., Verpoest, I., Peters, T., Roose, D., Parnas, R. S., Hoes, K., & Sol, H. (2004). Modelling of permeability of textile reinforcements: lattice Boltzmann method. *Composites Science and Technology. 64*, 1069-1080.

Bernsdorf, J., Brenner, G., & Durst, F. (2000). Numerical analysis of the pressure drop in porous media flow with lattice Boltzmann (BGK) automata. *Computer Physics Communications. 129*, 247-255.

Bousfield, D. W., & Karles, G. (2004). Penetration into three-dimensional porous structures. *Journal of Colloid and Interface Science, 270,* 396-405.

Bertrand, F., Gange, T., Desaulniers, E., Vidal D., & Hayes, R. E. (2004). Simulation of the consolidation of paper coating structures: probabilistic versus deterministic models. *Computers and Chemical Engineering, 28*, 2595-2604.

Carman, P. C. (1956). *Flow of Gases Through Porous Media*, London, UK: Butterworths Scientific Publications.

Clague, D. S., Kandhai, B. D., Zhang, R., & Sloot, P. M. A. (2000). Hydraulic permeability of (un)bounded fibrous media using the lattice Boltzmann method, *Physical Review E*, 61(1), 616-625.

Clague, D. S., & Phillips, R. J. (1997). A numerical calculation of the hydraulic permeability of three-dimensional disordered fibrous media, *Physics of Fluids, 9(6)*, 1562:1572.

Cook, R. A., & Hover, K. C. (1993). Mercury porosimetry of cement-based materials and associated correction factors, *ACI Materials Journal, March/April*, 152-161.

Desaulniers, E. (2003). *Modélisation de l'entassement de particules lors du procédés de couchage du papier*. M.Sc.A. Thesis, Ecole Polytechnique de Montréal, Canada.

Desaulniers, E., Bertrand, F., Leclaire, L.-A., & Vidal, D. (2005). Numerical modeling of granular flow with the discrete element method: application to pigment consolidation in paper coating process. *In proceedings of the 4th International Conference on CFD in the Oil and Gas*, Trondheim, Norway: Metallurgical & Process Industries.

Dullien, F. A. L. (1979). *Porous Media - Fluid Transport and Pore Structure*, New York, NY, USA: Academic Press.

Eksi, G., & Bousfield, D. W. (1997). Modeling of coating structure development. *Tappi Journal, 80(2)*, 125:135.

Fredrich, J. T., DiGiovanni, A. A., & Noble, D. R. (2006). Predicting macroscopic transport properties using microscopic image data", *Journal of Geophysical Research, 111*, B03201, doi:10.1029/2005JB003774.

Gane, P. A. C., Kettle, J. P., Matthews, G. P., & Ridgway, C. J. (1996). Void space structure of compressible polymer spheres and consolidated calcium carbonate paper-coating formulations. *Industrial and Engineering Chemistry Research, 3,* 1753-1764.

Gane, P. A. C., Schoelkopf, J., Spielmann, D. C., Matthews, G. P., & Ridgway, C. J. (2000). Fluid transport into porous coating structures: some novel findings. *Tappi Journal, 83(5)*, 77-78.

Guodong, J., Patzek, T. W., & Silin, D. B. (2004). Direct prediction of the absolute permeability of unconsolidated and consolidated reservoir rock. *In proceedings SPE Annual Technical Conference and Exhibition (SPE 90084)*, Richardson, TX, USA: Society of Petroleum Engineers, Inc.

Hayashi, H., Yamamoto, S., & Hyodo, S. (2003). Lattice-Boltzmann simulations of flow through nafion polymer membranes. *International Journal of Modern Physics B, 17(1/2)*, 135-138.

Hayashi, H., & Kubo, S. (2008). Computer simulation study on filtration of soot particles in diesel particulate filter. *Computers & Mathematics with Applications, 55(7)*, 1450-1460.

Hill, R. J., Koch, D. L., & Ladd, A. J. C., (2001). The first effects of fluid inertia on flows in ordered and random arrays of spheres. *Journal of Fluid Mechanics, 448*, 213-241.

Hiorns, T., & Nesbitt, T. (2003). Particle packing of blocky and platey pigments – a comparaison of computer simulations and experimental results. *In CD-ROM proceedings of 2003 TAPPI Advanced Coating Fundamentals Symposium*, Atlanta, GA, USA: TAPPI Press.

Humby, S. J., Biggs, M. J., & Tüzün, U. (2002). Explicit numerical simulation of fluids in reconstructed porous media, *Chemical Engineering Science, 57(11)*, 1955-1968.

Jia, X., & Williams, R. A. (2006). From microstructure of tablets and granules to their dissolution behaviour, *Dissolution Technologies, 13(2)*, 11-19.

Kang, Q., Zhang, D., & Chen, S. (2002) Unified lattice Boltzmann method for flow in multiscale porous media. *Physical Review E, 66(5)*, 56307.

Ladd, A. J. C. (1990). Hydrodymanic transport coefficients of random dispersions of hard spheres. *Journal of Chemical Physics, 93(5)*, 3484-3494.

Leskinen, A. M. (1987). Layered Structure in Model Coatings. *Tappi Journal, 70(12)*, 101-106.

Lyons, A. V., & Iyer, R. R. (2004). Use of particle packing modeling with lognormal particle size distributions to develop a strategy to improve blade coating runnability. *In proceedings of 2004 TAPPI Coating and Graphics Arts Conference and Exhibit*, CTG0438, Atlanta, GA, USA: TAPPI Press.

Manwart, C., Aaltosalmi, U., Koponen, A., Hilfer, R., & Timonen, J. (2002). Lattice-Boltzmann and finite-difference simulations for the permeability for three-dimensional porous media. *Physical Review E, 66(1)*, 016702.

Nourgaliev, R. R., Dinh, T. N., Theofanous T.G. & Joseph, D. (2003). On lattice Boltzmann modeling of phase transition in an isothermal non-ideal fluid. *International Journal of Multiphase Flow, 29(1)*, 117-169.

Perry, R. H., & Green, D. (1984), *Perry's Chemical Engineers' Handbook.* (6^{th} ed.) New York, NY, USA: McGraw Hill.

Petrasch, J., Meier, F., Friess, H., & Steinfeld, A. (2008). Tomography based determination of permeability, Dupuit-Forchheimer coefficient, and interfacial heat transfer coefficient in reticulate porous ceramics. *The International Journal of Heat and Fluid Flow, 29*, 315-326. (doi:10.1016/j.ijheatfluidflow.2007.09.001)

Pianet, G., Bertrand, F., Vidal, D., & Mallet, B. (2008). Modeling the compression of particle packings using the Discrete Element Method. *In proceedings of the 2008 TAPPI Advanced Coating Fundamentals Symposium*, Atlanta, GA, USA: TAPPI Press.

Pan, C., Luo, L., & C. Miller (2006). An evaluation of lattice Boltzmann schemes for porous medium flow simulation. *Computer & Fluids, 35(8/9)*, 898-909.

Pan, C., Hilpert, M., & C. Miller (2001). Pore-scale modeling of saturated permeabilities in random sphere packings. *Physical Review E, 64(6)*, 066702.

Quispe, J. R., & Toledo, P. G. (2004). Lattice-Boltzmann simulation of flow through two-dimensional particle sediments. *International Journal of Mineral Processing, 73*, 91-102.

Ridgway, C. J., Schoelkopf, J. & Gane, P. A. C. (2003). A new method for measuring the liquid permeability of coated and uncoated papers and boards. *Nordic Pulp and Paper Research Journal, 18(4)*, 379-383.

Sand, A., Toivakka, M., & Hjelt, T. (2006). Investigation of filter cake stability using numerical simulation technique. *In proceedings of the 2006 TAPPI Advanced Coating Fundamentals Symposium, p.279*, Atlanta, GA, USA: TAPPI Press.

Selomulya, C., Tran, T. M., Jia, X., & Williams, R. A. (2006). An integrated methodology to evaluate permeability from measured microstructures. *AIChE Journal, 52(10)*, 3394-3400.

Singh, M., & Mohanty, K. K. (2003). Dynamic modeling of drainage through three-dimensional porous materials. *Chemical Engineering Science, 58*, 1-18.

Succi, S. (2001). *The lattice Boltzmann Equation for Fluid Dynamics and Beyond.* Oxford, UK: Oxford Science Publications.

Sullivan, S. P., Gladden, L. F., & Johns, M. L. (2006). Simulation of power-law fluid flow through porous media using lattice Boltzmann techniques. *Journal of Non-Newtonian Fluid Mechanics, 133(2/3)*, 91-98.

Sullivan, S. P., Sederman, A. J., & Gladden, L. F. (2007). Verification of shear-thinning LB simulations in complex geometries. *Journal of Non-Newtonian Fluid Mechanics, 143(2/3)*, 59-63.

Tang, G. H., Tao, W. Q., & Lee, Y. L. (2005). Three-dimensional lattice Boltzmann model for gaseous flow in rectangular microducts and microscale porous media. *Journal of Applied Physics, 97*, 104918.

Toivakka, M., Eklund, D., & Bousfield, D. W. (1992). Simulation of pigment motion during drying. *In proceedings of 1992 TAPPI Coating Conference, p.403.* Atlanta, GA, USA: TAPPI Press.

Toivakka, M., Salminen, P., Chonde, Y., & Bousfield, D. W. (1997). Consolidation of particulate suspension - model study with plastic pigments. *In proceedings of 1997 TAPPI Advanced Coating Fundamentals Symposium, p.89.* Atlanta, GA, USA: TAPPI Press.

Toivakka, M., & Nyfors, K. (2001). Pore space characterization of coating layers, *Tappi Journal, 84(3)*, 1-16.

Tolke, J., Krafczyk, M., Schulz, M., & Rank, E. (2002). Lattice Boltzmann simulations of binary fluid flow through porous media. *Philosophical Transactions: Mathematical, Physical and Engineering Sciences, 360(1792)*, 535-545.

Van der Hoef, M. A, Beetstra R., & Kuipers J. A. M. (2005). Lattice-Boltzmann simulations of low-Reynolds-number flow past mono- and bidisperse arrays of spheres: results for the permeability and drag force. *Journal of Fluid Mechanics, 528*, 233-254.

Vidal, D., Zou, X., & Uesaka, T. (2003a). Modeling coating structure development using a Monte-Carlo deposition method - Part I: modeling methodology. *Tappi Journal, 2(4)*, 3-8.

Vidal, D., Zou, X., & Uesaka, T. (2003b). Modeling coating structure development using a Monte-Carlo deposition method - Part II: validation and case study.", *Tappi Journal., 2(5),*16-20.

Vidal, D., Zou, X., & Uesaka, T., (2004). Modelling coating structure development: Monte-Carlo deposition of particles with irregular shapes", *Nordic Pulp & Paper Research Journal, 19(4)*,442-449.

Vidal, D., & Bertrand, F. (2006). Recent progress and challenges in the numerical modeling of coating structure development, *In proceedings of the 2006 TAPPI Advanced Coating Fundamentals Symposium, p.241.* Atlanta, GA, USA: TAPPI Press.

Videla, A. R., Lin, C. L., & Miller, J. D. (2008). Simulation of saturated fluid flow in packed particle beds - The lattice-Boltzmann method for the calculation of permeability from XMT images, *Journal of Chinese Institute of Chemical Engineers, 39*, 117–128.

Wyllie, M. R. J., & Gregory, A. R. (1955). Fluid flow through unconsolidated porous aggregates - Effect of porosity and particle shape on Kozeny-Carman constants", *Industrial & Engineering Chemistry, 47(7)*, 1379-1388.

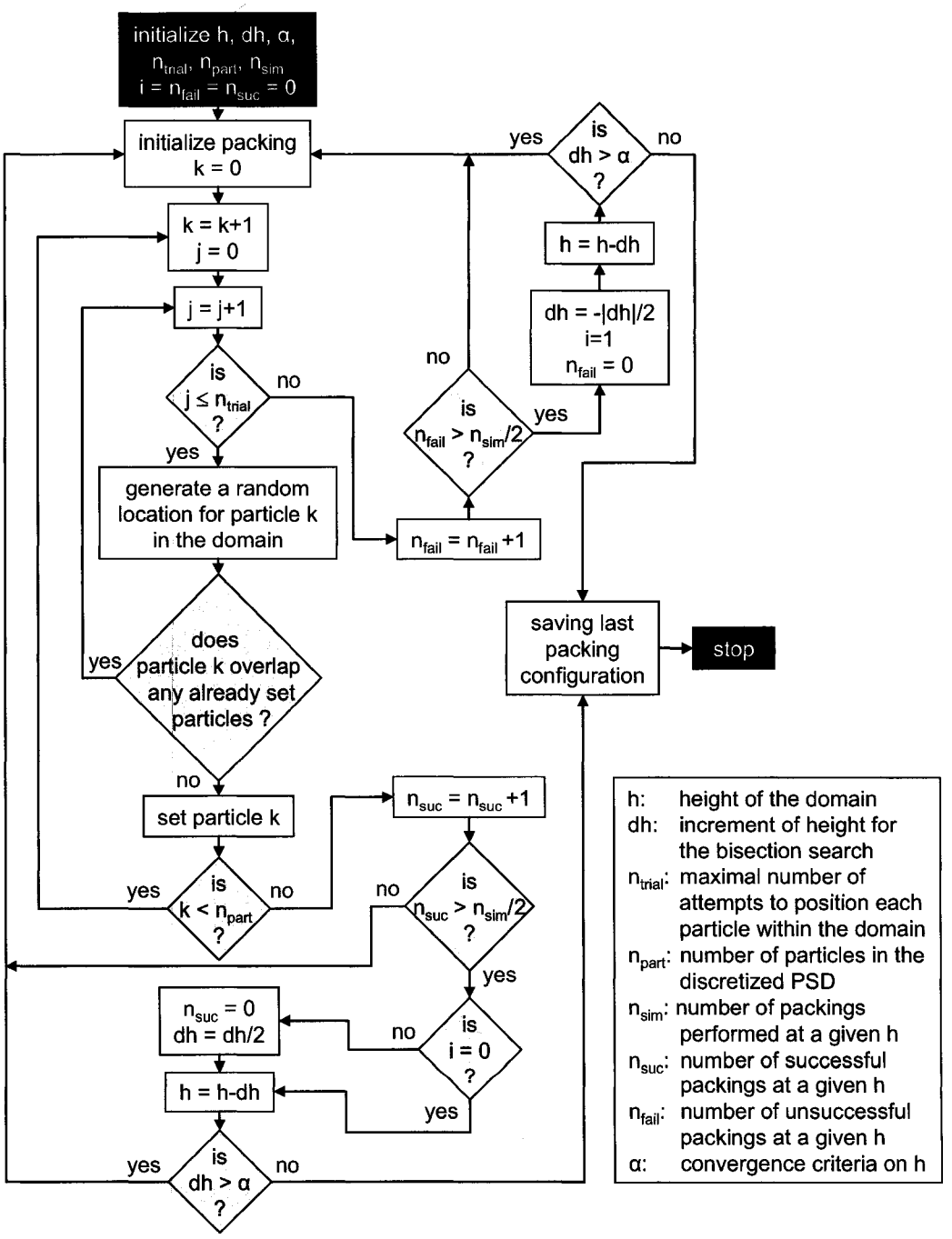
Xu, R.-N., & Jiang, P.-X. (2008). Numerical simulation of fluid flow in microporous media. *International Journal of Heat and Fluid Flow*, doi:10.1016/j.ijheatfluidflow.2008.05.005

Yamamoto, K., & Takada, N. (2006). LB simulation on soot combustion in porous media. *Physica A, 362*, 111-117.

Zeiser, T., Lammersa, P., Klemm, E., Li, Y. W., Bernsdorf, J., & Brenner, G. (2001). CFD-calculation of flow, dispersion and reaction in a catalyst filled tube by the lattice Boltzmann method. *Chemical Engineering Science, 56(4)*, 1697-1704.

Zeiser, T., Steven, M., Freund, H., Lammers, P., Brenner, G., Durst, F., & Bernsdorf, J. (2002). Analysis of the flow field and pressure drop in fixed-bed reactors with the help of lattice-Boltzmann simulations. *Philosophical Transactions: Mathematical, Physical and Engineering Sciences, 360(1792)*, 507-520.

6.7 APPENDIX: MONTE-CARLO PACKING (MCP) ALGORITHM



CHAPITRE 7

DISCUSSION GÉNÉRALE

Des combinaisons de stratégies informatiques ont amené à la conception de deux algorithmes parallèles, à savoir un algorithme de type *one-lattice* (Chapitre 4) et un autre de type *shift* (Chapitre 5) reposant sur une décomposition de domaine fondée sur une répartition équitable du vecteur de données et un transfert de données entre processeurs optimisé. Ceux-ci se sont montrés extrêmement avantageux en terme de performance parallèle et d'utilisation de la mémoire par rapport à des algorithmes faisant appel à de simples décompositions de domaine cartésiennes. Une analyse au moyen d'un modèle de performance théorique montre qu'ils procurent un équilibrage de tâches parfait. Bien que des décompositions de domaine basées sur des méthodes de partition de graphes pourraient en théorie procurer une meilleure évolutivité par la minimisation des communications, la taille des domaines étudiée (plusieurs milliards de nœuds fluides) proscrit l'utilisation de telles méthodes. L'algorithme de type *shift* avec temps de relaxation unitaire apparait supérieur à l'algorithme de type *one-lattice* proposé, mais est toutefois limité à des écoulements newtoniens à cause de son temps de relaxation constant. Des simulations d'écoulements de fluides dans des entassements compressés de sphères hautement polydisperses à l'aide de l'algorithme *one-lattice* proposé ont été réalisées et ont permis l'établissement d'une corrélation de Carman-Kozeny modifiée (Chapitre 6). Un tel résultat n'aurait probablement été possible ni avec d'autres méthodes de mécanique des fluides numérique, ni de façon expérimentale à cause de la complexité des expériences requises et de la précision des mesures et des calculs nécessaires. À notre connaissance, les simulations effectuées vont aussi bien au-delà par leur ampleur de celles précédemment publiées dans la littérature avec la méthode de Boltzmann sur réseau. En ce sens, les algorithmes développés dans cette thèse sont bien uniques.

CHAPITRE 8

CONCLUSIONS

Dans le cadre de la simulation d'écoulements de fluides monophasiques dans des milieux poreux saturés sur des architectures à mémoire distribuée ou hybride au moyen d'une méthode de Boltzmann sur réseau, deux algorithmes parallèles ont été conçus. L'un consiste en un algorithme de type *one-lattice* basé sur un schéma collision-propagation découplé et combiné à une structure de données vectorielle, une décomposition de domaine fondée sur une répartition équitable du vecteur de données et un transfert de données entre processeurs optimisé. L'autre algorithme de type *shift*, est basé sur un schéma collision-propagation fusionné simplifié (avec temps de relaxation unitaire), une structure de données vectorielle, une décomposition de domaine fondée sur une répartition équitable du vecteur de données et un transfert de données simplifié. Lorsque comparés à un algorithme *one-lattice* basé sur une simple décomposition de domaine par tranches et une structure de données matricielles sur deux grappes de calcul différentes, les deux algorithmes proposés se sont montrés extrêmement avantageux en terme de performance parallèle (permettant un équilibrage de tâches parfait) et d'utilisation de la mémoire. Une comparaison entre les deux algorithmes montrent que l'algorithme de type *shift* avec temps de relaxation unitaire est grandement supérieur à l'algorithme de type *one-lattice* proposé, mais limité à des écoulements de type newtonien à cause de son temps de relaxation constant.

L'efficacité à résoudre des problèmes concrets a été démontrée au moyen de la simulation d'écoulements de fluides dans des entassements compressés de sphères hautement polydisperses à l'aide de l'algorithme *one-lattice* proposé. Les perméabilités prédites numériquement se comparent très bien avec les données expérimentales et la corrélation de Carman-Kozeny. La précision des calculs rendue possible grâce à

l'utilisation des nouveaux algorithmes développés a permis de mettre toutefois en évidence une déviation de la perméabilité de celle prédite par Carman-Kozeny lorsque la polydispersité des sphères et le niveau de compression sont élevés. Nous avons donc pu proposer une modification à la corrélation de Carman-Kozeny tenant compte de la distribution de taille des sphères et de la compression du milieu.

Examinons maintenant en détail les principales contributions de notre travail et les recommandations pour des travaux futurs. Nous terminerons par une mise en perspective du travail.

8.1 PRINCIPALES CONTRIBUTIONS

Les principales contributions de ce travail sont de deux ordres, spécifiquement d'un point de vue génie informatique:

1. la combinaison judicieuse de diverses stratégies précédemment proposées dans la littérature (structure de données vectorielle, décomposition de domaine fondée sur une répartition vectorielle équitable, algorithmes *one-lattice* et *shift*, temps de relaxation unitaire) dans le but de réduire au maximum l'utilisation de la mémoire et d'accroître la performance parallèle;

2. l'amélioration des schémas de transfert de données entre processeurs dans le but de réduire la charge des communications d'une décomposition de domaine fondée sur une répartition vectorielle équitable;

3. la conception et la comparaison des deux algorithmes efficaces possédant une bonne évolutivité et portabilité, et la détermination de leurs conditions d'utilisation;

4. l'établissement de modèles théoriques de performance parallèle et d'utilisation de la mémoire dans le cas de domaines poreux permettant de juger de l'évolutivité d'algorithmes MBR;

5. l'établissement d'une métrique (Équation (4.16) de l'Article No.1) qui permet d'évaluer l'efficacité parallèle même si la taille du domaine étudié ne permet pas de mesurer le temps de calcul sur un seul processeur;

et d'un point de vue génie chimique:

6. la démonstration que la simulation de l'écoulement de fluide au travers de milieux extrêmement complexes est dorénavant possible avec une grande précision grâce à la combinaison de la méthode de Boltzmann sur réseau et d'algorithmes parallèles;

7. l'établissement d'une corrélation de Carman-Kozeny modifiée tenant compte du niveau de compression du milieu poreux et de la polydispersité des particules qui le constituent;

8.2 RECOMMANDATIONS

Comme nos résultats et les travaux récents de Mattila *et al.* [61] semblent le montrer, l'algorithme *shift* possède une meilleure performance séquentielle que l'algorithme *one-lattice.* Toutefois, l'algorithme *shift* n'a été utilisé qu'en combinaison avec un schéma collision-propagation fusionné simplifié. Il serait donc intéressant de le tester avec un schéma collision-propagation fusionné non-simplifié et de le comparer à notre implantation du *one-lattice.* De façon similaire, le tout nouvel algorithme *swap* pourrait aussi accroître les performances séquentielles.

Toujours pour améliorer les performances séquentielles, l'implantation d'une technique d'accélération de la convergence (voir Section 2.5.2) serait aussi intéressante à

tester sur les deux algorithmes développés. Aussi, le remplacement du rebond à mi-chemin comme condition frontière de mur sans glissement par une nouvelle approche permettant de faire varier le temps de relaxation devrait pouvoir nous permettre d'utiliser des pas de temps plus grands et de converger plus rapidement vers la solution.

D'un point de vue du génie chimique, le développement d'outils numériques aussi efficaces nous ouvre un vaste champ de possibilités de recherche. Parmi celles-ci, mentionnons l'impact sur la perméabilité de la forme des particules constituant le milieu poreux et de la ségrégation ou de l'agrégation des particules.

8.3 PERSPECTIVES

Plusieurs auteurs ont récemment utilisé MBR pour résoudre des problèmes d'ondes, de transfert de chaleur, de réaction diffusive et de transfert de matière [109-118]. Ils ont ainsi démontré que MBR constitue en fait un nouvel outil mathématique pour la résolution d'équations différentielles partielles et donc ne se limite pas aux seules équations de Navier-Stokes. L'idée fondamentale derrière MBR est de construire un modèle cinétique simplifié qui incorpore la physique essentielle des phénomènes micro ou mésoscopiques régissant les variables macroscopiques telles la vitesse, la température ou la concentration. La principale différence entre les différentes applications résident en fait dans l'utilisation des fonctions de distribution à l'équilibre appropriées (établies au moyen d'un développement multi-échelle de Chapman-Enskog) permettant de retrouver le comportement de l'équation différentielle partielle souhaitée. Les algorithmes parallèles présentés dans cette thèse revêtent donc un grand intérêt tant scientifique que technologique qui dépasse le seul sujet des écoulements de fluides dans les milieux poreux.

RÉFÉRENCES

[1] Hilfer, R., "Local Porosity Theory and Stochastic Reconstruction for Porous Media", K.R. Mecke and D. Stoyan (eds.): LNP 554, Springer-Verlag Berlin Heidelberg, 2000, pp.203-241.

[2] Dullien, F.A.L., *Porous Media – Fluid Transport and Pore Structure*, New York: Academic Press, 1979, p.396.

[3] Poulin, N., P.A. Tanguy, J. Aspler, et L. Larrondo, "Numerical and Physical Modeling of the Permeability of Paper to CMC and Coating Liquids", *Canadian Journal of Chemical Engineering*, vol. 75 , no. 5, pp.949-955, 1997.

[4] Petrasch, J., Meier, F., Friess, H., & Steinfeld, A., Tomography based determination of permeability, Dupuit-Forchheimer coefficient, and interfacial heat transfer coefficient in reticulate porous ceramics", The International Journal of Heat and Fluid Flow, 29, pp. 315-326, 2008. (doi:10.1016/j.ijheatfluidflow.2007.09.001)

[5] Torquato, S., *Random Heterogeneous Materials- Microstructure and Macroscopic Properties*, New York: Springer, 2002, p.701.

[6] Adler, P.M., *Porous Media - Geometry and Transports*, Stoneham, MA, USA: Butterworth-Heinemann Series in Chemical Engineering, 1992, p.544.

[7] Bear, J., *Dynamics of Fluids in Porous Media.* New York, NY, USA: Dover Publications Inc., 1972.

[8] Carman, P. C., *Flow of Gases Through Porous Media*, London, UK: Butterworths Scientific Publications, 1956.

[9] Nourgaliev, R.R., Dinh, T.N., Theofanous, T.G. & Joseph, D., "The lattice Boltzmann equation method: theoretical interpretation, numerics and implications", *International Journal of Multiphase Flow*, vol. 29 , no. 1 , 2003, pp.117-169.

[10] Bernsdorf, J., Durst, F. & Schäfer, M ., "Comparison of cellular automata and finite volume techniques for simulation of incompressible flows in complex geometries", *Int. J. Numer. Meth. Fluids*, vol. 29, pp.251-264, 1999.

[11] Kandhai, D., Vidal, D.J.-E., Hoekstra, A.G., Hoefsloot, H., Iedema, P. & Sloot, P.M.A., "A Comparison Between Lattice-Boltzmann and Finite-Element Simulations of Fluid Flow in Static Mixer Reactors", *Int. J. of Mod. Phys. C*, vol. 9, no. 8, pp.1123-1128, 1998.

[12] Videla, A. R., Lin, C. L., & Miller, J. D., "Simulation of saturated fluid flow in packed particle beds - The lattice-Boltzmann method for the calculation of permeability from XMT images", *Journal of Chinese Institute of Chemical Engineers*, vol. 39, pp. 117–128, 2008.

[13] Hayashi, H., & Kubo, S., "Computer simulation study on filtration of soot particles in diesel particulate filter", *Computers & Mathematics with Applications*, vol. 55, no. 7, pp. 1450-1460, 2008.

[14] Yamamoto, K., & Takada, N., "LB simulation on soot combustion in porous media", *Physica A*, vol. 362, pp. 111-117, 2006.

[15] Sullivan, S. P., Gladden, L. F. & Johns, M. L., "Simulation of power-law fluid flow through porous media using lattice Boltzmann techniques", *Journal of Non-Newtonian Fluid Mechanics*, vol. 133, no. 2/3, pp. 91-98, 2006.

[16] Selomulya, C., Tran, T. M., Jia, X. & Williams, R. A., "An integrated methodology to evaluate permeability from measured microstructures", *AIChE Journal*, vol. 52, no. 10, pp. 3394-3400, 2006.

[17] Pan, C., Luo, L. & Miller C., "An evaluation of lattice Boltzmann schemes for porous medium flow simulation", *Computer & Fluids*, vol. 35, no. 8/9, pp. 898-909, 2006.

[18] Fredrich, J.T., DiGiovanni, A.A. & Noble, D.R., "Predicting macroscopic transport properties using microscopic image data", *Journal of Geophysical Research*, vol. 111, B03201, doi:10.1029/2005JB003774, 2006.

[19] Van der Hoef, M.A., Beetstra R. & Kuipers J.A.M., "Lattice-Boltzmann simulations of low-Reynolds-number flow past mono- and bidisperse arrays of spheres: results for the permeability and drag force", *Journal of Fluid Mechanics*, vol. 528, pp. 233-254, 2005.

[20] Tang, G.H., Tao, W.Q. & Lee, Y.L., "Three-dimensional lattice Boltzmann model for gaseous flow in rectangular microducts and microscale porous media", *Journal of Applied Physics*, vol. 97, 104918, 2005.

[21] Quispe, J.R. & Toledo, P.G., "Lattice-Boltzmann simulation of flow through two-dimensional particle sediments", *International Journal of Mineral Processing*, vol. 73, pp. 91-102, 2004.

[22] Guodong, J., Patzek, T.W. & Silin, D.B., "Direct prediction of the absolute permeability of unconsolidated and consolidated reservoir rock", *In proceedings SPE Annual Technical Conference and Exhibition (SPE 90084)*, Richardson, TX, USA: Society of Petroleum Engineers, Inc., 2004

[23] Belov, E.B., Lomov, S.V., Verpoest, I., Peters, T., Roose, D., Parnas, R.S., Hoes, K. & Sol, H., "Modelling of permeability of textile reinforcements: lattice Boltzmann method", *Composites Science and Technology*, vol. 64, pp. 1069-1080, 2004.

[24] Aaltosalmi, U., Kataja M., Koponen A., Timonen J., Goel A., Lee G. & Ramaswamy, S., "Numerical analysis of fluid flow through fibrous porous materials", *Journal of Pulp and Paper Science*, vol. 30, no. 9, pp. 251-255, 2004.

[25] Hayashi, H., Yamamoto, S. & Hyodo, S., "Lattice-Boltzmann simulations of flow through nafion polymer membranes", *International Journal of Modern Physics B*, vol. 17, no. (1/2), pp. 135-138, 2003.

[26] Humby, S.J., Biggs, M. J. & Tüzün, U., "Explicit numerical simulation of fluids in reconstructed porous media", *Chemical Engineering Science*, vol. 57, no. 11, pp. 1955-1968, 2002.

[27] Kang, Q., Zhang, D. & Chen, S., "Unified lattice Boltzmann method for flow in multiscale porous media", *Phys. Rev. E*, vol. 66, no. 5 , part. 2, pp.56307(1-11) , 2002.

[28] Manwart, C., Aaltosalmi, U., Koponen, A., Hilfer, R. & Timonen, J., "Lattice-Boltzmann and finite-difference simulations for the permeability for three-

dimensional porous media", *Phys. Rev. E,*. vol. 66 , no. 1 , part. 2, pp.016702(1-11), 2002.

[29] Tolke, J., Krafczyk, M., Schulz, M. & Rank, E., "Lattice Boltzmann simulations of binary fluid flow through porous media", *Phil. Trans. R. Soc. Lond. A.*, vol. 360 , no. 1792, pp.535-545, 2002.

[30] Zeiser, T., Steven, M., Freund, H., Lammers, P., Brenner, G., Durst, F. & Bernsdorf, J., "Analysis of the flow field and pressure drop in fixed-bed reactors with the help of lattice-Boltzmann simulations", *Phil. Trans. R. Soc. Lond. A*, vol. 360, pp.507-520, 2002.

[31] Hill, R. J., Koch, D. L. & Ladd, A. J. C., "The first effects of fluid inertia on flows in ordered and random arrays of spheres", *Journal of Fluid Mechanics*, vol. 448, pp. 213-241, 2001.

[32] Kotila S. & Haataja, J., "CSC Report on Scientific Computing 1999–2000", Scientific Computing Ltd., Finland, 2001, p.224 (http://www.csc.fi/reports/cr99-00/).

[33] Pan, C., Hilpert, M. & Miller, C., "Pore-scale modeling of saturated permeabilities in random sphere packings", *Physical Review E*, vol. 64, no. 6, 066702, 2001.

[34] Zeiser, T., Lammersa, P., Klemm, E., Li, Y. W., Bernsdorf, J. & Brenner, G., "CFD-calculation of flow, dispersion and reaction in a catalyst filled tube by the lattice Boltzmann method", *Chemical Engineering Science*, vol. 56, no. 4, pp. 1697-1704, 2001.

[35] Bernsdorf, J., Brenner, G. & Durst, F., "Numerical analysis of the pressure drop in porous media flow with lattice Boltzmann (BGK) automata", *Comp. Phys. Comm.*, vol. 129, pp.247-255, 2000.

[36] Clague, D. S., Kandhai, B. D., Zhang, R., & Sloot, P. M. A., "Hydraulic permeability of (un)bounded fibrous media using the lattice Boltzmann method", *Physical Review E*, vol. 61, no. 1, pp. 616-625, 2000.

[37] Martys, N.S., "Diffusion in partially-saturated porous materials", *Materials and Structures*, vol. 32, pp.555-562, 1999.

[38] Martys, N.S., Hagedorn, J.G., Goujon, D. & Devaney, J.E., "Large Scale Simulations of Single and Multi-component Flow in Porous Media", SPIE Conference on Developments in X-Ray Tomography II, Denver, 1999, pp.205-213.

[39] Koponen, A., Kandhai, D., Hellen, E., Alava, M., Hoekstra, A., Kataja, M. & Niskanen, K., "Permeability of Three-Dimensional Random Fiber Webs", *Phys. Rev. Letters*, vol. 80, no. 4, pp.716-719, 1998.

[40] Spaid, M.A.A. & Phelan, F.R., "Lattice-Boltzmann methods for modeling microscale flow in fibrous porous media", *Phys. Fluids*, vol. 9, no. 9, pp.2468-2474, 1997.

[41] Martys, N.S. & Chen, H., "Simulation of multicomponent fluids in complex three-dimensional geometries by the lattice Boltzmann method", *Phys. Rev. E*, vol. 53, no. 1, pp.743-750, 1996.

[42] Qi, D. & Uesaka, T., "Numerical experiments on paper-fluid interaction – permeability of a three-dimensional anisotropic fibre network", *J. Mat. Sci.*, vol. 31, pp.4865-4870, 1996.

[43] Ferréol. B. & Rothman, D., "Lattice-Boltzmann Simulations of Flow Through Fontainbleau Sandstone", *Trans. Porous Media*, vol. 20, pp. 3-20, 1995.

[44] Martys, N.S., Torquato, S. & Bentz, D.P., "Universal scaling of fluid permeability for sphere packings", *Phys. Rev. E*, vol. 50, no. 1, pp.403-408, 1994.

[45] Auzerais, F.M., Dunsmuir, J., Ferréol, B.B., Martys, N., Olson, J., Ramakrishnan, T.S., Rothman, D.H. & Schwartz, L.M., "Transport in sandstone: a study based on thre dimensional microtomography", *Geophys. Res. Lett.*, vol. 23, no. 7, pp.705-708, 1996.

[46] Rothman, D.H. & Zaleski, S., *Lattice-Gas Cellular Automata – Simple Models of Complex Hydrodynamics*, Cambridge University Press, 1997, p.297.

[47] Succi, S., *The lattice Boltzmann Equation for Fluid Dynamics and Beyond*, Oxford Science Publications, 2001, p.288.

[48] Wolf-Gladrow, D.A., *Lattice-Gas Cellular Automata and Lattice Boltzmann Models – An Introduction*, Berlin: Springer, 2000, p.308.

[49] Kandhai, D., "Large Scale Lattice-Boltzmann Simulations – Computational Methods and Applications", Thèse de doctorat, Université d'Amsterdam, 1999, p.154.

[50] Gardner, M., "Mathematical Games", *Scientific American*, October, pp.120-123, 1967.

[51] Wilkinson, B. & Allen, M., *Parallel Programming – Techniques and Applications using Networked Workstations and Parallel Computers*, New Jersey: Prentice Hall, 1999, p.431.

[52] Nagel, K. & Schreckenberg, M., "A cellular automaton model for freeway traffic", *J. Phys. I France 2*, pp.2221, 1992.

[53] Frisch, U., Hasslacher, B. & Pomeau, Y., "Lattice-gas automata for the Navier-Stokes equatons", *Phys. Rev. Lett.*, vol. 56, pp.1505-1508, 1986.

[54] McNamara, G. & Zametti, G., "Use of the Boltzmann equation to simulate lattice gas automata", *Phys. Rew. Lett.*, vol. 61, pp.2332, , 1988.

[55] Bhatnager, P.L., Gross, E.P. & Krook, M., "Model for collision processes in gases", *Phys. Rev.*, vol. 94, pp.511, 1954.

[56] He, X. & Luo, L.-S., "Theory of the lattice Boltzmann method: From the Boltzmann equation to the lattice Boltzmann equation", *Phys. Rev. E*, vol. 56, no. 6, pp. 6811-6817, 1997.

[57] He, X. & Luo, L.-S., "A priori derivation of the lattice Boltzmann equation", *Phys. Rev. E*, vol. 55, no. 6, part A, pp. R6333 , 1997.

[58] Teng, S.L., Chen, Y. & Ohashi, H., "Lattice Boltzmann simulation of multiphase fluid flows through the total variation diminishing with artificial compression scheme", *Int. J. Heat Fluid Flow*, vol. 21, pp.112-121, 2000.

[59] McNamara, G.R., Garcia, A.L. & Adler, B.J. "Stabilization of thermal lattice Boltzmann models", *J. Stat. Phys.*, vol. 81, pp.395-408, 1995.

[60] Maier, R., Bernard, R.S. & Grunau, D.W., "Boundary conditions for the lattice Boltzmann method", *Phys. Fluids*, vol. 8, no. 7, pp. 1788-1801, 1996.

[61] Mattila K., Hyväluoma J., Timonen J. & Rossi, T., "Comparison of implementations of the lattice-Boltzmann method" *Comput Math Appl*, vol. 55, no. 7, pp. 1514-1524, 2008.

[62] Schulz M., Krafczyk M., Tolke J. & Rank, E., "Parallelization strategies and efficiency of CFD computations in complex geometries using lattice Boltzmann methods on high performance computers", In: Breuer M., Durst F., Zenger C., editors. *High performance scientific and engineering computing. Berlin: Springer Verlag*, 2002, p.115-122.

[63] Massaioli, F. & Amati, G., "Achieving high performance in a LBM code using OpenMP", in: *The Fourth European Workshop on OpenMP* (EWOMP 2002), Roma, September 18–20, 2002.

[64] Pohl, T., Kowarschik, M., Wilke, J., Iglberger, K. & Rüde, U., "Optimization and profiling of the cache performance of parallel lattice Boltzmann codes", *Parallel Process Lett*, vol. 13, no. 4, pp.549-560, 2003.

[65] Mattila, K., Hyväluoma, J., Rossi, T., Aspnäs, M. & Westerholm, J., "An efficient swap algorithm for the lattice Boltzmann method", *Comp Phys Comm*, vol. 176, pp. 200-210, 2007.

[66] Martys, N.S. & Hagedorn, J.G. "Multiscale modeling of fluid transport in heterogeous materials using descrete Boltzmann methods", *Mater Struct*, vol. 35, pp. 650-659, 2002.

[67] Dupuis, A., & Chopard, B., "An object oriented approach to lattice gas modeling", *Future Gener Comput Syst*, vol. 16, no. 5, pp. 523-532, 2000.

[68] Pan, C., Prins, J.F. & Miller, C.T., "A high-performance lattice Boltzmann implementation to model flow in porous media", *Comput Phys Commun*, vol. 158, pp.89-105, 2004.

[69] Wang, J., Zhang, X., Bengough, A.G. & Crawford, J.W. "Domain-decomposition method for parallel lattice Boltzmann simulation of incompressible flow in porous media", *Phys. Rev. E*, vol. 72, pp. 016706-11, 2005.

[70] Wellein, G., Zeiser, T., Hager, G. & Donath, S., "On the single processor performance of simple lattice Boltzmann kernels", *Comput Fluids*, vol. 35, pp. 910-919, 2006.

[71] Murphy, S. "Performance of Lattice Boltzmann Kernels", M.Sc. dissertation, University of Edinburgh, UK, 2005.

[72] Bernachi, M. & Succi, S., "Accelerated Lattice Boltzmann Scheme for Steady-state Flows", *Int. J. Mod. Phys. B*, vol. 17, no. 1, p.1-6, 2003.

[73] Bernachi, M. & Succi, S. & Chen, H.,"Accelerated Lattice Boltzmann Schemes for Steady-state Flow Simulations", *J. Sci. Comp.*, vol. 16, no. 2, pp.135-144, 2001.

[74] Kandhai, D., Koponen, A., Hoekstra, A. & Sloot, P.M.A., "Iterative momentum relaxation for fast lattice-Boltzmann simulations", *Future Generation Comp. Syst.*, vol.18, pp.89-96, 2001.

[75] Freudiger, S., Hegewald, J. & Krafczyk, M., "A parallelization concept for a multi-physics lattice Boltzmann prototype based on hierarchical grids", *Progr Comput Fluid Dynam Int J*, vol. 8, no. 1-4, pp.168-178, 2008.

[76] Yu, D., Mei, R. & Shyy, W., "A multi-block lattice Boltzmann method for viscous fluid flows", *Int. J. Numer. Meth. Fluids*, vol. 39, pp.99-120, 2002.

[77] Kandhai D., Soll, W., Chen, S., Hoekstra, A. & Sloot, P., "Finite-Difference Lattice-BGK methods on nested grids", *Computer Phys. Comm.*, vol. 129, pp. 100-109, 2000.

[78] Lin, C.-L. & Lai, Y.G., "Lattice Boltzmann method on composite grids", *Phys. Rev. E*, vol. 62, no. 2, pp.2219-2225, 2000.

[79] Mazzocco, F., Arrighetti, C., Bella, G., Spagnoli, L. & Succi, S., "Multiscale lattice Boltzmann schemes: a preliminary application to axial turbomachine flow simulations", *Int. J. Modern Phys.*, vol. 11, no. 2, pp.233-245, 2000.

[80] Mei, R. & Shyy, W., "On the Finite Difference-Based Lattice Boltzmann Method in Curvilinear Coordinates", *J. Comp. Phys.*, vol. 143, pp.426-448, 1998.

[81] Cao, N., Chen, S., Jin, S. & Martinez, D., "Physical symmetry and lattice symmetry in the lattice Boltzmann method", *Phys. Rev. E*, vol. 55, no. 1, pp.R21-R24, 1997.

[82] Chew, Y.T., Shu, C. & Peng, Y., "On Implementation of Bondary Conditions in the Application of Finite Volume Lattice Boltzmann Method", *J. Stat. Phys.*, vol. 107, no. 1/2, pp.539-556, 2002.

[83] Peng, G., Xi, H., Duncan, C. & Chou, S.-H., "Finite volume scheme for the lattice Boltzmann method on unstructured meshes", *Phys. Rev. E*, vol. 59, no. 4, pp.4675-4682, 1999.

[84] Peng, G., Xi, H. & Chou, S.-H., "On boundary conditions in the finite volume lattice Boltzmann method on unstructured meshes", *Int. J. Modern Phys. C*, vol. 10, no. 6, pp.1003-1016, 1999.

[85] Xi, H., Peng, G. & Chou, S.-H., "Finite-volume lattice Boltzmann schemes in two and three dimensions", *Phys. Rev. E*, vol. 60, no. 3, pp.3380-3388, 1999.

[86] Nannelli, F. & Succi, S., "The Lattice Boltzmann Equation on Irregular Lattices", *J. Stat. Phys.*, vol. 68, no. 3/4, pp.401-407, 1992.

[87] He, X. & Luo, L.-S., "A priori derivation of the lattice Boltzmann equation", *Phys. Rev. E*, vol. 55, no. 6, pp. R6333-R6336, 1997.

[88] He, X., Luo, L.-S. & Dembo, M., "Some Progress in Lattice Boltzmann Method. Part I. nonuniform Mesh Grids", *J. Comp. Phys.*, vol. 129, pp. 357-363, 1996.

[89] Chew, Y.T., Shu, C. & Niu, X.D., "Simulation of unsteady incompressible flows by using Taylor series expansion- and least squares-based lattice Boltzmann method", *Int. J. Modern Phys.*, vol. 13, no. 6, pp.719-738, 2002.

[90] Shu, C., Niu, X.D. & Chew, Y.T., "Taylor-series expansion and least-squares-based lattice Boltzmann method : Two-dimensional formulation and its applications", *Phys. Rev. E*, vol. 65, no. 036708, pp.1-13, 2002.

[91] Shu, C., Chew, Y.T. & Niu, X.D., "Least-squares-based lattice Boltzmann method : A meshless approach for simulation of flows with complex geometry", *Phys. Rev. E*, vol. 64, no. 045701, pp.1-4, 2001.

[92] Karlin, I.V., Succi, S. & Orszag, S., "Lattice Boltzmann Method for Irregular Grids", *Phys. Rev. E*, vol. 82, no. 26, pp.5245-5248, 1999.

[93] Lai, Y.G., Lin, C.-L. & Huang, J., "Accuracy and efficiency study of lattice Boltzmann method for steady-state flow simulations", *Numer. Heat Trans.*, Part B, vol. 39, pp.21-43, 2001.

[94] Axner, L., Bernsdorf, J., Zeiser, T., Lammers, P., Linxweiler, J. & Hoekstra, A.G., "Performance evaluation of a parallel sparse lattice Boltzmann solver", *J Comp Physics*, vol. 227, pp.4895-4911, 2008.

[95] Wang, J., Zhang, X., Bengough, A.G. & Crawford, J.W., "Domain-decomposition method for parallel lattice Boltzmann simulation of incompressible flow in porous media", *Phys. Rev. E*, vol. 72, pp. 016706-11, 2005.

[96] Suviola, T., "Parallelization of a lattice Boltzmann Suspension Flow Solver,", PARA 2002, LNCS 2367, Springer-Verlag, 2002, pp.603-610.

[97] Satofuka, N. & Nishioka, T., "Parallelization of lattice Boltzmann method for incompressible flow computations", *Computational Mechanics*, vol. 23, pp.164-171, 1999.

[98] Kandhai, D., Koponen, A., Hoekstra, A.G., Kataja, M., Timonen, J. & Sloot, P.M.A. "Lattice-Boltzmann hydrodynamics on parallel systems", *Comput. Phys. Comm.*, vol. 111, pp.14-26, 1998.

[99] Amati, G., Succi, S. & Piva, R., "Massively parallel lattice-Boltzmann simulation of turbulent channel flow", *Int. J. Modern Phys. C*, vol. 8, no. 4, pp.869-877, 1997.

[100] Noble, D.R., Georgiadis, J.G. & Buckius, R.O., "Comparison of accuracy and performance for lattice Boltzmann and finite difference simulations of steady viscous flow", *Int. J. Num. Meth. Fluids*, vol. 23, pp.1-18, 1996.

[101] Punzo, G., Massaioli, F. & S. Succi, "High-resolution lattice-Boltzmann computing on the IBM SP1 scalable parallel computer", *Comput. Phys.*, vol. 8, no. 6, pp.705-711, 1994.

[102] Diekmann, R., Monien, B. & Preis, R., "Load Balancing Strategies for Distributed Memory Machines", Karsch/Monien/Satz (éd.), *Multi-Scale Phenomena and Their Simulation*, World Scientific, 1997, pp.255-266.

[103] Biswas, R., Das, S.K., Harvey, D.J. & et Oliker, L., "Parallel dynamic load balancing strategies for adaptative irregular applications", *Appl. Math. Model.*, vol. 25, pp.109-122, 2000.

[104] Karypis, G. & Kumar, V., "Multilevel k-way partitioning scheme for irregular graphs", *J. Parallel Distributed Computing*, vol. 481, no. 1, p.96-129, 1998.

[105] Walshaw, C. & Cross, M., "Parallel optimisation algorithms for multilevel mesh partitioning", *Parallel Computing*, vol. 26, pp.1635-1660, 2000.

[106] Karypis Lab, http://glaros.dtc.umn.edu/gkhome/node/419, consulté novembre 2008

[107] Zou, Q. & He, X., "On pressure and velocity boundary conditions for the lattice Boltzmann BGK model", *Phys. Fluids*, vol. 9, no. 6, p. 1591-1598, 1997.

[108] Chen, S., Martinez, D. & Mei, R. "On boundary conditions in lattice Boltzmann methods", *Phys. Fluids*, vol. 8, no. 9, pp. 2527-2536, 1996.

[109] Guangwu, Y., "A Lattice Boltzmann Equation for Waves", *J. Comp. Phys.*, vol. 161, pp.61-69, 2000.

[110] Ho, J.-R., Kuo, C.P. & Jiaung, W.-S., "Study of heat transfer in multilayered structure within the framework of dual-phase-lag heat conduction model using lattice Boltzmann method", *Int. J. Heat Mass Trans.*, vol. 46, pp.55-69, 2003.

[111] Jiaung, W.-S., Ho, J.-R. & Kuo, C.P., "Lattice Boltzmann method for the heat conduction problem with phase change", *Numer. Heat Trans.*, Part B, vol. 39, pp.167-187, 2001.

[112] Prakash, G.S., Reddy, S.S., Das, S.K., Sundararajan, T. & Seetharamu, K.N., "Numerical modeling of microscale effects in conduction for different thermal boundary conditions", *Numer. Heat Trans.*, Part A, vol. 38, pp.513-532, 2000.

[113] van der Sman, R.G.M., Ernst, M.H. & Berkenbosch, A.C., "Lattice Boltzmann scheme for cooling of packed cut flowers", *Int. J. Heat Mass. Trans.*, vol. 43, pp.577-587, 2000.

[114] Yoshino, M. & Inamuro, T., "Lattice Boltzmann simulations for flow and heat/mass transfer problems in a three-dimensional porous structure", *Int. J. Numer. Meth. Fluids*, vol. 43, pp. 183-198, 2003.

[115] van der Sman, R.G.M., & Ernst, M.H., "Diffusion Lattice Boltzmann Scheme on a Orthorhombic Lattice", *J. Stat. Phys.*, vol. 94, no. 1/2, pp.203-217, 1999.

[116] Wolf-Gladrow, D., "A Lattice Boltzmann Equation for Diffusion", *J. Stat. Phys.*, vol. 79, no. 5/6, pp.1023-1032, 1995.

[117] Elton, B.H., Levermore, C.D. & Rodrigue, G.H., "Lattice Boltzmann methods for some 2-D nonlinear diffusive equations: Computational results", H. Kaper éd., *Proceedings of the Workshop on Asymptotic Analysis and Numerical Solution of PDEs*, Argonne National Laboratory, New York: Marcell Dekker, 1990, pp.197-214.

[118] Qian, Y.H. & Orszag, S.A., "Scalings in Diffusion-driven Reaction A+B → C: Numerical Simulations by Lattice BGK Models", *J. Stat. Phys.*, vol. 81, pp. 237-253, 1995.

ANNEXE A

VITESSES DES RÉSEAUX D2Q9 ET D3Q15

Les réseaux D2Q9 et D3Q15 possèdent respectivement 9 et 15 vecteurs vitesses de discrétisation différents tels qu'illustrés à la Figure 2.5. Mathématiquement, les vitesses peuvent être exprimées comme suit :

1. D2Q9 :

$$\mathbf{e}_0 = [0,0]$$

$$\mathbf{e}_i = \frac{\delta_x}{\delta_t}\left[\cos\left(\frac{\pi(i-1)}{2}\right), \sin\left(\frac{\pi(i-1)}{2}\right)\right], \quad i = 1,2,...,4, \tag{A.1}$$

$$\mathbf{e}_i = \sqrt{2}\frac{\delta_x}{\delta_t}\left[\cos\left(\frac{\pi(i-5)}{2}+\frac{\pi}{4}\right), \sin\left(\frac{\pi(i-5)}{2}+\frac{\pi}{4}\right)\right], \quad i = 5,6,...,8.$$

2. D3Q15 :

$$\mathbf{e}_i = \frac{\delta_x}{\delta_t}\left[E(1,i+1), E(2,i+1), E(3,i+1)\right], \quad i = 0,1,...,14$$

$$\text{avec } E = \begin{bmatrix} 0 & 1 & -1 & 0 & 0 & 0 & 0 & 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 \\ 0 & 0 & 0 & 1 & -1 & 0 & 0 & 1 & -1 & 1 & -1 & -1 & 1 & -1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & -1 & 1 & -1 & -1 & 1 & 1 & -1 & -1 & 1 \end{bmatrix} \tag{A.2}$$

ANNEXE B

FONCTIONS DE DISTRIBUTION À L'ÉQUILIBRE POUR D2Q9 ET D3Q15

Les fonctions de distributions à l'équilibre donnent le bon comportement hydrodynamique pour un fluide (presque) incompressible dans la limite de faibles nombres de Mach et de Knudsen. Nous avons d'après Nourgaliev *et al.* [9]:

Pour i = 0 :

$$f_0^{eq}(\mathbf{x},t) = w_0\rho\left[1-\frac{\mathbf{u}\cdot\mathbf{u}}{2}\left(3\left(\frac{\delta_t}{\delta_x}\right)^2-\frac{1-w_0}{c_s^2}\right)\right] \tag{B.1}$$

et pour $i \neq 0$:

$$f_i^{eq}(\mathbf{x},t) = w_i\rho\left[1-\frac{1}{2}\frac{\mathbf{u}\cdot\mathbf{u}}{c_s^2}+\frac{(\mathbf{e}_i\cdot\mathbf{u})}{c_s^2}+\frac{3}{2}\left(\frac{\delta_t}{\delta_x}\right)^2\frac{(\mathbf{e}_i\cdot\mathbf{u})^2}{c_s^2}\right] \tag{B.2}$$

où les définitions de w_i (le poids de la population i) et c_s (la vitesse du son du réseau) varient en fonction du type de réseau utilisé et du poids w_0 de la population au repos. Nous avons:

<u>Pour un réseau D2Q9:</u>

$$w_i = \frac{1-w_0}{5},\ \text{pour } i=1:4; \quad w_i = \frac{1-w_0}{20},\ \text{pour } i=5:8; \tag{B.3}$$

$$c_s = \sqrt{\frac{3(1-w_0)}{5}}\frac{\delta_x}{\delta_t}, \tag{B.4}$$

et le poids de la population au repos est souvent choisi comme $w_0 = \frac{4}{9}$ ou $\frac{1}{2}$.

Pour un réseau D3Q15:

$$w_i = \frac{1-w_0}{7}, \text{ pour } i = 1:6; \quad w_i = \frac{1-w_0}{56}, \text{ pour } i = 7:14; \tag{B.5}$$

$$c_s = \sqrt{\frac{3(1-w_0)}{7}}\frac{\delta_x}{\delta_t}, \tag{B.6}$$

et le poids de la population au repos est souvent choisi comme $w_0 = \frac{1}{8}$ ou $\frac{2}{9}$.

Toutefois, pour un comportement de fluide purement incompressible, des fonctions de distribution légèrement modifiées ont été proposées pour un D2Q9 [107] :

$$f_i^{eq}(\mathbf{x},t) = w_i\left[\rho + 3(\mathbf{e}_i\cdot\rho\mathbf{u}) + \frac{9}{2}(\mathbf{e}_i\cdot\rho\mathbf{u})^2 - \frac{3}{2}\rho\mathbf{u}\cdot\rho\mathbf{u}\right] \tag{B.7}$$

ANNEXE C

CONDITIONS FRONTIÈRES

Jusqu'au milieu des années 90, les conditions frontières ont constitué la pierre d'achoppement de MBR, tout particulièrement pour les simulations tridimensionnelles. En effet, les chercheurs avaient du mal à trouver des conditions frontières de mur qui soient parfaitement conservatives dans le cas tridimensionnel. Comme pour la méthode de MBR en général, le développement des conditions frontières adéquates a aussi été quelque peu heuristique. Nous ne nous lancerons donc pas ici dans une dissertation détaillée des différentes conditions frontières disponibles et de leur développement. Nous référons plutôt le lecteur aux références [46,107-108]. Toutefois, nous donnerons ici un aperçu des conditions qui pourront être d'utilité dans le cadre de la simulation des écoulements dans les milieux poreux.

C.1 CONDITIONS DE MUR

Étant donné que notre milieu poreux est rempli d'éléments solides, la première question qui nous vient probablement à l'esprit est: comment fait-on pour imposer une condition de non-glissement à la paroi de ses éléments (conditions de mur ou de vitesse nulle à la paroi) ? L'idée en théorie est relativement simple : les populations pointant vers une paroi solide sont réfléchies vers l'intérieur du domaine. On donne le nom de conditions de rebond à ce genre de conditions frontières. Toutefois, il y a plusieurs façons d'exécuter ce rebond et certaines mènent à de meilleurs résultats que d'autres. Entre

autres, on a les conditions de « rebond sur grille » et de « rebond à mi-chemin ». Examinons-les plus en détails.

C.1.1 Rebond sur grille

Dans le cas du rebond sur grille, la frontière physique se situe exactement sur des nœuds de la grille. Les populations y sont tout simplement inversées[44]. Pour un nœud tel qu'illustré à la Figure C.1 et dont les coordonnées sont (x,y), on obtient donc la transformation suivante :

$$\begin{pmatrix} f_7(x,y,t+\delta_t) \\ f_4(x,y,t+\delta_t) \\ f_8(x,y,t+\delta_t) \end{pmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{pmatrix} f_5(x,y,t) \\ f_2(x,y,t) \\ f_6(x,y,t) \end{pmatrix}$$

La précision de cette condition est souvent créditée comme étant du premier ordre à cause du caractère unilatéral de l'opérateur de propagation à la frontière.

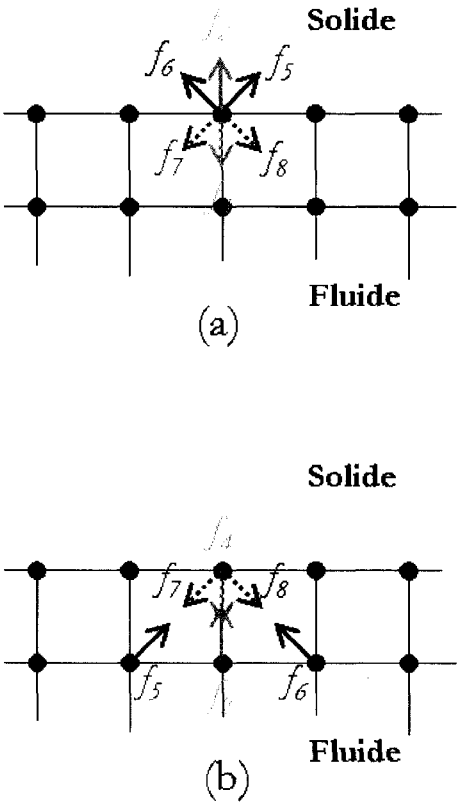


Figure C.1 – Illustration des conditions de rebond sur grille (a) et à mi-chemin pour un réseau D2Q9 (flèches pleines avant le rebond et flèches pointillées après).

[44] En pratique, seules les populations rentrantes le sont, les autres ne servant pas.

C.1.2 Rebond à mi-chemin

Dans le cas du rebond à mi-chemin, la frontière physique se situe en fait à mi-chemin entre l'avant dernière ligne de nœuds (derniers nœuds réellement « fluides ») et la dernière ligne (nœuds « solides »). Ce sont les populations des nœuds fluides voisins qui sont réfléchies sur un nœud solide donné. Le rebond est donc ici un peu plus complexe et en conséquence il est illustré à la Figure C.1 (une image vaut mille mots!!). La transformation suivante s'applique donc :

$$\begin{pmatrix} f_7(x,y,t+\delta_t) \\ f_4(x,y,t+\delta_t) \\ f_8(x,y,t+\delta_t) \end{pmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{pmatrix} f_5(x-1,y-1,t) \\ f_2(x,y-1,t) \\ f_6(x+1,y-1,t) \end{pmatrix} \tag{C.2}$$

Si la frontière se trouve en pratique à mi-chemin, c'est que l'aller-retour entre un nœud solide et un nœud fluide est de $2\times\delta_x$ et le temps pour l'effectuer seulement de δ_t. Ce type de rebond revient à implanter un schéma de discrétisation en temps et en espace centrée dont la précision est du deuxième ordre[45]. Notons finalement qu'une attention particulière doit être apportée aux nœuds de coin pour assurer les lois de conservation (pour plus de détails, voir pp.114-116 de [46]).

C.2 IMPOSITION D'UN GRADIENT DE PRESSION

Si l'on veut imposer un gradient de pression à travers un milieu poreux, on peut penser à imposer à l'entrée un profil de vitesse et à la sortie un gradient de vitesse de zéro

[45] Toutefois, comme le soulignent certains chercheurs, cette distinction faite au sujet de la précision entre les deux types de rebond ne serait pas très juste. La technique de rebond sur grille pourrait être en fait amenée à une précision du second ordre [46].

ou, encore, imposer des densités différentes aux deux extrémités. Bien que ces deux possibilités soient faisables, on leur préfère très souvent dans la littérature l'imposition de conditions frontières périodiques à l'entrée et à la sortie, combinées à l'ajout d'une quantité de mouvement au moyen d'une force externe.

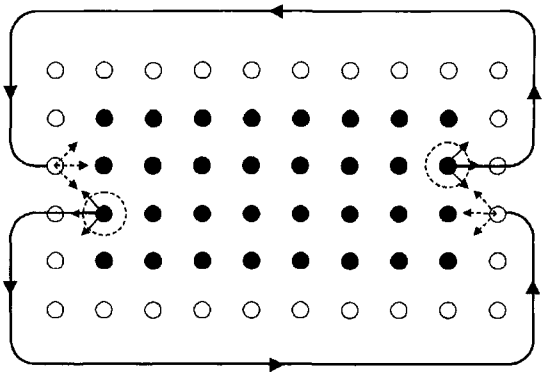


Figure C.2 – Conditions frontières périodiques (cercles pleins = nœuds du domaine, cercles vides = nœuds tampons).

Les conditions périodiques sont simples (Figure C.2) :

- toutes les populations pointant vers l'extérieur à la sortie sont réinjectées à l'entrée comme populations pointant vers l'intérieur;
- et toutes les populations pointant vers l'extérieur à l'entrée sont réinjectées à la sortie comme populations pointant vers l'intérieur.

En ce qui concerne la force externe $\|\mathbf{f}\|$, elle est en pratique, en chaque nœud et à chaque itération (étape (8e) de l'algorithme C&P) :

- ajoutée dans les directions ayant une composante non-nulle positive par rapport au sens de l'écoulement induit par la perte de charge;
- et retranchées dans les directions miroir des précédentes de façon à garantir la conservation de la masse et de la quantité de mouvement.

Mathématiquement, ceci s'exprime par :

$$f_i(\mathbf{x},t) = f_i(\mathbf{x},t) + \frac{\mathbf{e}_i \cdot \mathbf{f}}{\|\mathbf{e}_i \cdot \mathbf{f}\|}\|\mathbf{f}\| \quad \forall i \,|\, \mathbf{e}_i \cdot \mathbf{f} \neq 0 \tag{C.3}$$

$$\text{avec } \|\mathbf{f}\| = \frac{1}{b\, c_s^2}\frac{\|\Delta P\|}{L} \tag{C.4}$$

où b le nombre de directions affectées par la force externe[46] (6 avec un D2Q9, 10 avec un D3Q15), L la distance sur laquelle s'applique la perte de charge ΔP. Cependant, l'Équation (2.20) donnant le profil de pression doit maintenant être remplacée par l'équation suivante :

$$P(\mathbf{x},t) = c_s^2\left(\rho(\mathbf{x},t) - \langle\rho(\mathbf{x},t)\rangle\right) - \frac{\|\Delta P\|}{L}(x_1 - L) \qquad \text{pour } 0 \le x_1 \le L. \tag{C.5}$$

Cette correction est due au fait que l'on force les densités à l'entrée et à la sortie à être égales par l'utilisation de conditions périodiques, donc il ne peut pas y avoir en pratique de différence de pression. Cependant, les fluctuations locales de pression dans le domaine dues, par exemple, à un obstacle (c'est-à-dire la pression dynamique) sont adéquatement calculées par cette méthode.

Pour l'imposition des conditions frontières, notons qu'il est commun et pratique d'utiliser une rangée de nœuds tampons (voir Figure C.2) tout autour du domaine de façon à faciliter leur implantation dans l'algorithme général de la méthode [46]. Finalement, la séquence d'imposition des conditions frontières a son importance et doit s'exécuter dans un ordre précis [60] :

1. Conditions périodiques (propagation vers les nœuds tampons);
2. Conditions de rebond (vers les nœuds tampons ou solides);
3. Imposition de la force externe;
4. Conditions de densités prescrites;
5. Conditions de vitesses prescrites.

[46] C'est-à-dire les vitesses de discrétisation ayant une composante non nulle dans la direction de la force externe.

ANNEXE D

STABILITÉ DE L'ALGORITHME C&P

Rappelons tout d'abord que la méthode de MBR-BGK repose sur des hypothèses de faibles nombres de Mach et de Knudsen, c'est-à-dire explicitement :

$$\mathrm{Ma} = \frac{U}{c_s} << 1 \text{ et } \mathrm{Kn} << 0,01 \tag{D.1}$$

où U une vitesse caractéristique qui définit ici la plus grande vitesse que peut supporter le réseau avant de devenir instable.

De plus, étant donné le schéma de différentiation explicite de l'algorithme, les conditions habituelles de Courant-Friedrichs-Lewy (CFL) doivent être vérifiées de façon à garantir la stabilité de la méthode. Nous avons donc le nombre de Courant convectif :

$$\mathrm{Co_u} = \frac{U\delta_t}{\delta_x} < 1 \quad \Rightarrow U < \frac{\delta_x}{\delta_t} \tag{D.2}$$

et le nombre de Courant diffusif ou visqueux :

$$\mathrm{Co_v} = \frac{\nu\delta_t}{\delta_x^2} = \frac{\left(\tau^* - \frac{1}{2}\right)\delta_t^2 c_s^2}{\delta_x^2} < 1. \tag{D.3}$$

Si les hypothèses de base (Équations (D.1)) sont respectées, la première condition est en fait toujours garantie car le nombre de Courant convectif est égal au nombre de Mach à une constante près ($\mathrm{Co_u} \cong \mathrm{Ma} << 1$)[47]. La deuxième condition permet quant à elle

[47] $c_s = \beta\delta_x/\delta_t$ où β est une constante qui dépend du réseau (voir Annexe B).

de définir de façon plus précise la borne supérieure de la fenêtre de stabilité pour τ^*. La stabilité « numérique » est donc garantie si :

$$\frac{1}{2} < \tau^* < \frac{1}{\beta^2} + \frac{1}{2}, \tag{D.4}$$

ce qui, pour les grilles telles que définies à l'Annexe B, donne respectivement pour D2Q9 et D3Q15 des valeurs de 3,5 et 3,17 pour la borne supérieure. Toutefois, la stabilité numérique ne garantie pas la stabilité en pratique de la méthode! En effet, pour peu que la méthode ne soit pas strictement conservative, les erreurs résultantes pourraient mener à une perte de stabilité. Cependant, tel que démontré par [46], MBR-BGK est une méthode strictement conservative, aux erreurs d'arrondi près dans les calculs en point flottant[48]. La méthode est-elle alors garantie de converger ? Oui, mais... seulement dans le régime laminaire! Lorsque le nombre de Reynolds (Re) devient trop grand, le terme non-linaire d'inertie des équations de Navier-Stokes ($\mathbf{u} \cdot \nabla \mathbf{u}$) devient trop important et engendre une déstabilisation de l'algorithme. Si l'on définit comme critère de stabilité le Reynolds local suivant

$$\mathrm{Re}_x = \frac{U\delta_x}{\nu} <\sim 10-100 \tag{D.5}$$

on remarque que, comme pour toute discrétisation, la stabilité de l'algorithme peut-être améliorée en réduisant la taille de maille de la grille. Toutefois, ceci va avoir pour effet d'augmenter le temps de relaxation (Équation (2.17)) qui, à son tour, pourrait sortir de la fenêtre de stabilité (Équation (D.4)). Pour compenser, on est alors obligé de réduire quadratiquement le pas de temps (δ_t). On voit bien ainsi que MBR devient vite très chère numériquement parlant pour des Reynolds élevés, limitant donc en pratique son

[48] En effet, la propagation est parfaitement conservative puisque elle implique seulement le transfert des populations d'un nœud à un autre et la collision l'est aussi, aux erreurs d'arrondi près, car les lois de conservation sont encodées explicitement au niveau des fonctions d'équilibre local. Toutefois, des conditions frontières non conservatives pourraient déstabiliser la méthode dans son ensemble. On veillera donc encore une fois à les choisir avec précautions.

utilisation aux écoulements laminaires (Reynolds faibles à modérés) [9], à moins d'avoir un ordinateur extrêmement puissant!

ANNEXE E

MÉTHODES DE MBR-BGK ADAPTATIVES

E.1 MÉTHODES MULTI-ÉCHELLES OU À GRILLES EMBARQUÉES

Plusieurs chercheurs ont proposé l'utilisation de grilles locales embarquées [75-79]. Celles-ci permettent une discrétisation locale plus ou moins fine en fonction des conditions du problème. Elles sont classifiées selon une structure hiérarchique des niveaux de discrétisation, c'est-à-dire que les différents niveaux de grille opèrent à des échelles de temps et de longueur différents et doivent donc être résolu itérativement de façon séquentielle, de la grille la plus grossière vers la plus fine (Figure E.1). En général, le facteur de raffinement entre une grille donnée et la grille de niveau hiérarchique inférieur est de 2 et le couplage est réalisé entre elles par interpolation aux interfaces. Malheureusement, la résolution séquentielle hiérarchique des divers niveaux de

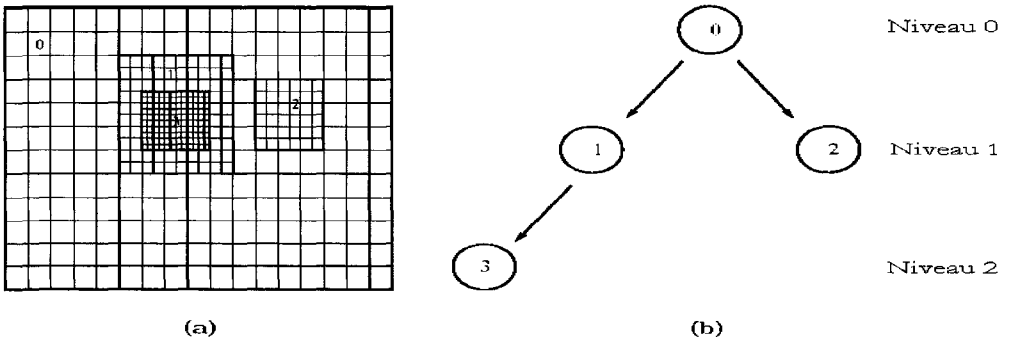


Figure E.1 – Exemple de grilles locales embarquées (a) et de la hiérarchie de résolution résultante (b) [77].

discrétisation se prête très mal à la parallélisation par décomposition de domaine ou tout du moins à l'équilibrage de tâches statique sur les processeurs. En effet, il faut pouvoir garantir, d'un processeur à l'autre, des densités équivalentes ou similaires des niveaux hiérarchiques, chose peu aisée s'il en est.

E.2 MÉTHODES MBR AUX DIFFÉRENCES FINIES

Cette méthode (MBR-DF) nous ramène en fait en arrière dans notre développement de MBR et plus précisément à l'équation de Boltzmann discrète (Équations 2.11) [80,81]. Cette dernière est cette fois-ci exprimée dans un repère eulérien en coordonnées cartésiennes[49] (supposons **g**=0 et deux dimensions pour simplifier l'exposé) :

$$\frac{\partial f_i}{\partial t} + e_{ix}\frac{\partial f_i}{\partial x} + e_{iy}\frac{\partial f_i}{\partial y} = -\frac{f_i - f_i^{eq}}{\tau} \qquad \text{(E.1)}$$

où e_{ix} et e_{iy} sont les composantes respectivement en x et y des vitesses $\mathbf{e}_i$. On applique ensuite des transformations sur les coordonnées qui vont permettre de raffiner, grossir ou distordre localement le réseau, soit $x=x(\xi)$ et $y=y(\eta)$ (Figure E.2). L'Équation E.1 devient donc :

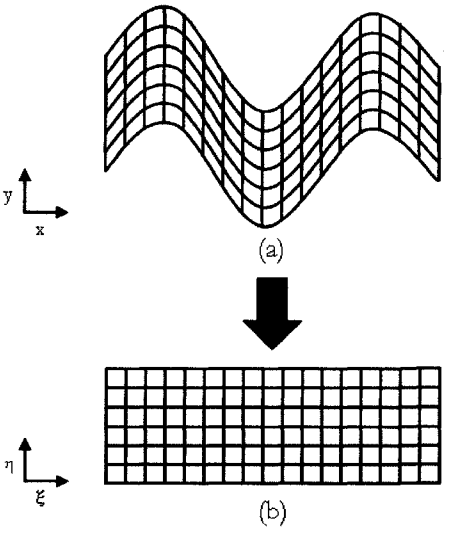


Figure E.2 – Concept de MBR-DF : transformation de coordonnées cartésiennes (a) en coordonnées curvilignes (b).

$$\frac{\partial f_i}{\partial t} + \frac{e_{ix}}{h_x}\frac{\partial f_i}{\partial \xi} + \frac{e_{iy}}{h_y}\frac{\partial f_i}{\partial \eta} = -\frac{f_i - f_i^{eq}}{\tau} \qquad \text{(E.2)}$$

[49] La technique peut aussi être développée, par exemple, à partir de coordonnées polaires.

où $h_x=dx/d\xi$ et $h_y=dy/d\eta$. Tel que démontré dans [80], l'Équation E.2 discrétisée en temps et en espace se ramène donc à :

$$\frac{f_i(\mathbf{x},t+\delta_t)-f_i(\mathbf{x},t)}{\delta_t}+\frac{e_{ix}}{h_x}\frac{\partial f_i(\mathbf{x},t)}{\partial\xi}+\ldots$$
$$\ldots+\frac{e_{iy}}{h_y}\frac{\partial f_i(\mathbf{x},t)}{\partial\eta}=-\frac{\left[f_i(\mathbf{x},t+\delta_t)-\left(2f_i^{eq}(\mathbf{x},t)-f_i^{eq}(\mathbf{x},t-\delta_t)\right)\right]}{\tau} \qquad \text{(E.3)}$$

où $\frac{\partial f_i(\mathbf{x},t)}{\partial\xi}$ et $\frac{\partial f_i(\mathbf{x},t)}{\partial\eta}$ sont discrétisés au temps t à l'aide d'une différentiation spatiale finie centrée ou à décentrage avant. Cela implique donc que cette méthode conserve l'avantage du caractère explicite de la méthode de MBR-BGK. Toutefois, autant pour une géométrie simple il serait relativement facile de définir des transformations à appliquer, autant pour une géométrie tels les milieux poreux ces transformations seraient ardues pour ne pas dire impossible à déterminer[50]. En conséquence, nous espérons pouvoir trouver parmi les autres méthodes un terreau plus propice à une discrétisation aisée des milieux poreux.

E.3 MÉTHODES MBR AUX VOLUMES FINIS

Différentes variantes de MBR aux volumes finis (MBR-VF) ont été proposées au cours des ans [82-86]. Nous nous contenterons ici de décrire brièvement la plus aboutie d'entre elles [85]. Comme son nom l'indique, celle-ci consiste en fait tout simplement à discrétiser l'équation de Boltzmann discrète par une méthode de volumes finis qui repose sur un maillage formé de triangle (ou tetrahèdre) ou de quadrilatère (ou hexaèdre). Un exemple de volume fini centré sur un point P du maillage est présenté à la Figure E.3. Un

[50] On pourrait toutefois imaginer utiliser pour se faire des réseaux de neurones à fonctions de base radiale. Mais ceci est une autre histoire…

bilan de population sur un élément de volume mène à la mise à jour des populations au point P :

$$f_i(\mathrm{P},\mathrm{t}+\delta_\mathrm{t}) = f_i(\mathrm{P},\mathrm{t}) + \frac{\delta_\mathrm{t}}{\mathrm{A_P}}\left(\sum_\mathrm{j}\Omega_{\mathrm{i,j}} + \sum_\mathrm{j}\Phi_{\mathrm{i,j}}\right) + \alpha\mathbf{e_i}\cdot\mathbf{f} \tag{E.4}$$

où A_p est l'aire totale du volume fini autour du point P, $\alpha = 1/\sum_i \mathbf{e}_{ix}^2 = 1/\sum_i \mathbf{e}_{iy}^2$ est un coefficient pour la force externe **f** (voir Annexe C) et les deux sommations sont respectivement les sommations des termes de collision et de flux sur tous les triangles composant le volume fini (ABP, BCP, ..., LAP). Par exemple, pour le triangle ABP, l'intégration des collisions à travers ses arêtes donne :

$$\begin{aligned}\Omega_{\mathrm{i,ABP}} &= -\int_{\mathrm{ABP}}\frac{1}{\tau}\left(f_\mathrm{i} - f_\mathrm{i}^{\mathrm{eq}}\right)\mathrm{d}\sigma \\ &= -\frac{\mathrm{A_{ABP}}}{3\tau}\left[\left(f_\mathrm{i}(\mathrm{P}) - f_\mathrm{i}^{\mathrm{eq}}(\mathrm{P})\right) + \left(f_\mathrm{i}(\mathrm{A}) - f_\mathrm{i}^{\mathrm{eq}}(\mathrm{A})\right) + \left(f_\mathrm{i}(\mathrm{B}) - f_\mathrm{i}^{\mathrm{eq}}(\mathrm{B})\right)\right]\end{aligned} \tag{E.5}$$

où les valeurs aux points A et B sont interpolées linéairement grâce aux valeurs aux nœuds environnants du maillage (P, P_1 et P_2). De façon similaire, nous intégrons le flux :

$$\Phi_{\mathrm{i,ABP}} = \int_{\mathrm{ABP}}\mathbf{e}_\mathrm{i}\cdot\nabla f_\mathrm{i}\,\mathrm{d}\sigma = \left(\mathbf{e}_\mathrm{i}\cdot\mathbf{n}_{\mathrm{AB}}\right)\mathrm{l}_{\mathrm{AB}}\frac{\left(f_\mathrm{i}(\mathrm{A}) + f_\mathrm{i}(\mathrm{B})\right)}{2} + \mathrm{I_s} \tag{E.6}$$

où $\mathbf{n}_{AB}$ est le vecteur normal à l'arête AB, l_{AB} est la longueur de AB et I_s représente les flux sur les arêtes internes (AP et BP) qui au moment de la sommation de l'Équation (E.5) viendront s'annuler avec ceux des autres triangles[51].

[51] Toutefois, les I_s ne s'annulent pas un à un dans le cas d'un volume fini à la frontière (voir [85]).

Pour les conditions frontières de mur, les populations pointant vers l'intérieur du domaine sont calculées par rebond (voir Annexe C). Pour d'autres conditions frontières, le lecteur est invité à consulter les références [82,84].

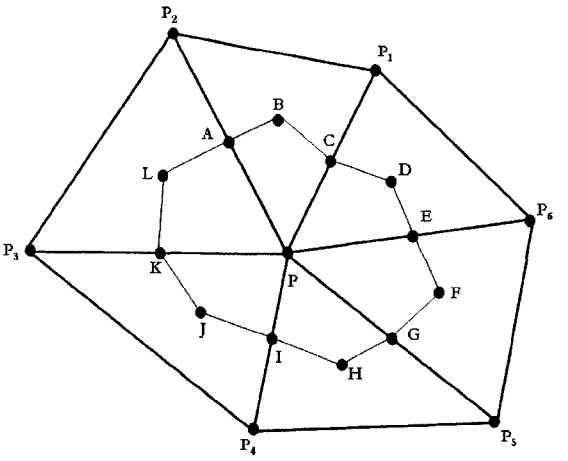


Figure E.3 – Volume fini (polyèdre A à L) centré sur le point P d'un maillage triangulaire. Les sommets du volume fini sont soit aux centres des arêtes, soit aux barycentres des triangles formés avec les points adjacents du maillage (P_1 à P_6).

En résumé, cette méthode conserve l'inhérente localité et donc la facilité à la parallélisation de la méthode MBR standard tout en permettant le maillage adaptatif. Toutefois, comme on peut le constater, le nombre de calculs requis pour les mises à jour des populations à chaque itération est significativement plus important. De l'espace mémoire supplémentaire est aussi éventuellement nécessaire pour stocker l'information sur les mailles. De plus, les méthodes de volumes finis requièrent en général que le maillage se conforme à la géométrie du milieu (Figure E.4). On retombe alors dans le problème de la génération de maillages adaptatifs pour des géométries complexes.

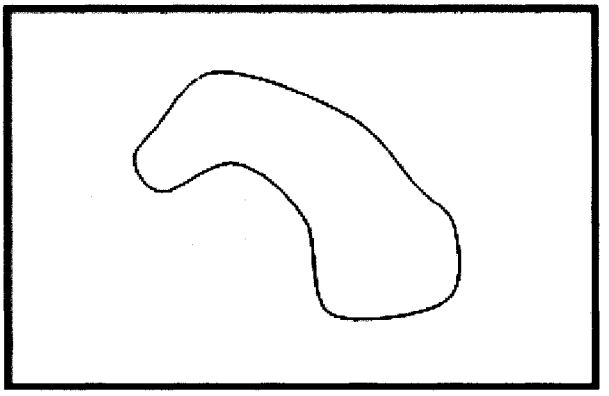


Figure E.4 – Maillage adaptatif conformant.

E.4 MÉTHODES D'INTERPOLATION

La dernière méthode repose sur le constat que rien n'oblige en pratique le couplage des discrétisations espace-temps et vitesse. L'idée est donc toute simple : à partir de points (ou positions d'interpolation) discrétisant le domaine (par exemple, P à la Figure E.5), la phase de collision est exécutée et les populations sont propagées vers des points de l'espace (P′) ne correspondant pas nécessairement à d'autres positions d'interpolation. En conséquence, pour pouvoir procéder à la prochaine itération, les populations sur les points doivent être interpolées à partir des populations « propagées » avoisinantes (c'est-à-dire A′, B′, C′, D′, E′, F′ et P′ pour obtenir les populations en P). Cette interpolation peut être réalisée de diverses manières, linéairement [87-88] ou au moyen d'une expansion en série de Taylor du deuxième ordre [89-91]. Ce genre de méthode est souvent dénommé méthode « sans maillage » ou « sans grille », ceci constituant dans une certaine mesure un abus de langage car la méthode nécessite tout de même un ensemble de points d'interpolation discrétisant le domaine. Toutefois, l'établissement de cet ensemble de points n'est régi par aucun critère général de structure ou d'uniformité, permettant ainsi au besoin une discrétisation flexible adaptative du domaine[52].

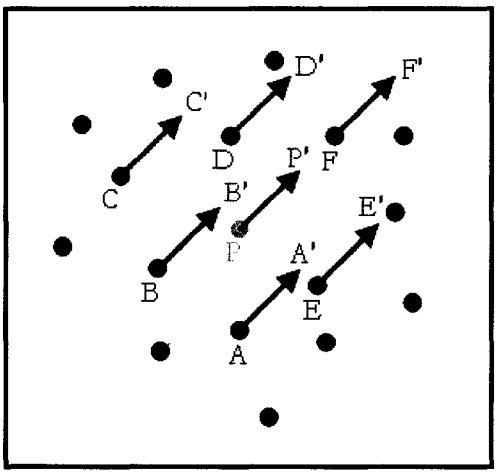


Figure E.5 – Propagation d'une population au point P dans une des directions de la discrétisation en vitesse, et populations « propagées » (A′, B′, C′, D′, E′, F′ et P′) utilisées pour l'interpolation subséquente en P.

Parmi toutes les méthodes examinées, la méthode proposée par Shu *et al.* [89-91] semble à première vue être la méthode la plus flexible pour la résolution de l'équation de Boltzmann dans les milieux poreux. Nous ne rentrerons pas dans le détail du

[52] En pratique, certains chercheurs préfèrent utiliser une grille structurée (mais pas nécessairement uniforme) de façon à simplifier la gestion des interpolations.

développement des équations qui au demeurant est relativement simple mais un peu long. Nous préférons plutôt présenter ici les grandes lignes à suivre pour résoudre l'équation de Boltzmann. Par conséquent, la mise à jour des populations est donnée par[53] :

$$f_i(\mathbf{x}_0, t+\delta_t) = \sum_{k=1}^{M+1} a_{1,k}\left[f_i(\mathbf{x}_{k-1}, t) + \left(f_i^{eq}(\mathbf{x}_{k-1}, t) - f_i(\mathbf{x}_{k-1}, t)\right)/\tau^* \right] \tag{E.7}$$

où $\mathbf{x}_0$ sont les coordonnées du point P et M est le nombre de points servant à l'interpolation (au minimum 6 points d'interpolation sont nécessaires). Les $a_{1,k}$ représentent quant à eux les éléments de la première rangée de la matrice [A] définie comme suit :

$$[A] = \left([S]^T [S] \right)^{-1} [S]^T \tag{E.8}$$

où :

$$[S] = \begin{bmatrix} 1 & \Delta x_0 & \Delta y_0 & (\Delta x_0)^2/2 & (\Delta y_0)^2/2 & \Delta x_0 \Delta y_0 \\ 1 & \Delta x_1 & \Delta y_1 & (\Delta x_1)^2/2 & (\Delta y_1)^2/2 & \Delta x_1 \Delta y_1 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & \Delta x_M & \Delta y_M & (\Delta x_M)^2/2 & (\Delta y_M)^2/2 & \Delta x_M \Delta y_M \end{bmatrix}_{(M+1)\times 6} \tag{E.9}$$

avec :

$$\begin{aligned} &\Delta x_0 = e_{ix}\delta_t, \quad \Delta y_0 = e_{iy}\delta_t, \\ &\Delta x_i = x_i + e_{ix}\delta_t - x_0, \quad \Delta y_i = y_i + e_{iy}\delta_t - y_0 \quad \text{pour } i = 1,2,\ldots,M. \end{aligned} \tag{E.10}$$

où les (x_i, y_i) sont les coordonnées des points d'interpolation voisins de P (c'est-à-dire A,B,.., F). Fort heureusement, les $a_{1,k}$ sont déterminés une fois pour toute au début de la

[53] Lorsque une force externe est utilisée, l'Équation (E.7) peut être remplacée par :

$$f_i(\mathbf{x}_0, t+\delta_t) = \sum_{k=1}^{M+1} a_{1,k}\left[f_i(\mathbf{x}_{k-1}, t) + \left(f_i^{eq}(\mathbf{x}_{k-1}, t) - f_i(\mathbf{x}_{k-1}, t)\right)/\tau^* + \frac{\mathbf{e}_i \cdot \mathbf{f}}{\|\mathbf{e}_i \cdot \mathbf{f}\|}\|\mathbf{f}\| \right] \quad \forall i | \mathbf{e}_i \cdot \mathbf{f} \neq 0$$

simulation car il n'implique que les coordonnées des points d'interpolation qui sont a priori connus d'avance.

La méthode de Shu *et al.* repose sur une interpolation du deuxième ordre par une expansion en série de Taylor combinée à une résolution par une méthode de moindres carrés[54]. Nous la nommerons dorénavant en abrégé : MBR-ITMC. Les conditions frontières sont implantées de façon identique à MBR standard et, en ce qui concerne l'algorithme de résolution, il est en presque tous points similaire à l'algorithme C&P présenté en page 26 pour MBR-BGK standard. Les seuls changements à y apporter concernent les étapes (2) et (8f) :

2. Discrétisation du domaine par un ensemble de points d'interpolation, définition de δ_t et du nombre de points d'interpolation M, détermination des M plus proches voisins de chaque point, définition des conditions frontières et calcul des coefficients $a_{l,k}$ via l'Équation (E.8);

8. Itération jusqu'à convergence du champ de vitesse;

    f. À partir des M points voisins, interpolation des nouvelles populations en tout point du réseau (où $\mathbf{x}=\mathbf{x}_0$) et dans les n_d directions du réseau :

$$f_i(\mathbf{x}_0, t+\delta_t) = \sum_{k=1}^{M+1} a_{l,k}\, f_i^*(\mathbf{x}_{k-1}, t), \quad i = 0,1,...,(n_d-1); \tag{E.11}$$

Comme on peut le voir, le nouvel algorithme n'a rien perdu de la localité originale de MBR standard tout en permettant une discrétisation adaptative. Toutefois, l'adaptativité a un certain coût du point de vue des ressources informatiques :

[54] La méthode de moindres carrés est introduite pour éviter les problèmes reliés à une matrice [S] qui serait singulière ou mal conditionnée et pour permettre de généraliser la méthode à plus de 6 nœuds d'interpolation. En effet, lorsque plus de 6 nœuds sont utilisés, le système d'équations à résoudre devient surdéterminé, d'où l'emploi d'une méthode de moindres carrés de façon à minimiser l'erreur commise [91].

- accroissement des besoins en mémoire pour stocker les coefficients $a_{i,k}$ en chaque point d'interpolation, soit en double précision (M+1)×N×8 octets, où N est le nombre de points d'interpolation[55].

- accroissement du temps de calcul dû à la préparation initiale du maillage et de la liste de voisins, et au calcul initial des coefficients $a_{i,k}$ (étape (2)), mais surtout au remplacement du transfert de données en mémoire de l'étape (8f) originale (page 27) par un calcul comprenant (M+1) opérations.

Les coûts sub-mentionnés semblent à première vue bien en deçà de ceux de MBR-VF. En fait, MBR-ITMC semble être parmi toutes les méthodes adaptatives étudiées à la fois la plus simple, la plus flexible et la plus efficace.

Finalement, pour être tout à fait exhaustif, mentionnons qu'un cinquième type de méthodes adaptatives a été proposé par le groupe de Succi [92]. Celle-ci repose sur la construction de populations locales à l'équilibre (f_i^{eq}) sujettes à une liste étendue de contraintes de moment qui prennent en compte la géométrie du réseau. Cette méthode requiert la résolution par des solveurs itératifs en chaque nœud du réseau irrégulier et à chaque pas de temps d'un système linéaire d'équations de taille n_d (n_d étant le nombre de vitesses discrètes). En plus d'être à première vue moins « parallèle » que d'autres méthodes, elle n'a pas pour le moment suscité l'engouement des chercheurs.

[55] Soit un accroissement de mémoire minimum de 56 octets/nœud, ce qui n'est pas négligeable.